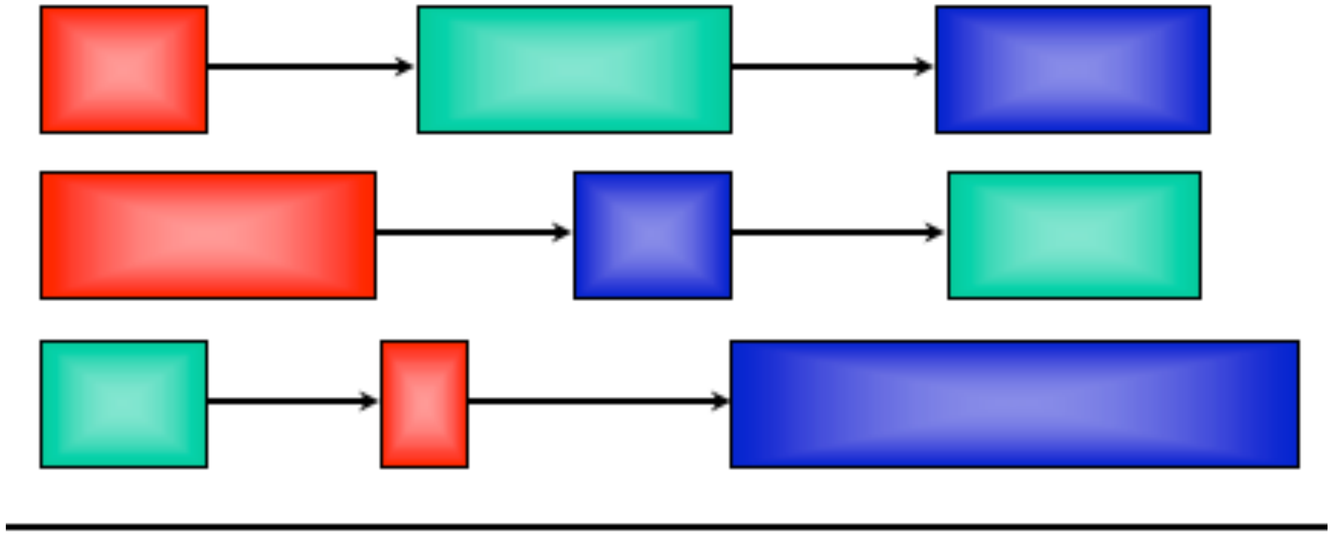
UNIVERSITY OF TORONTO | Engineering

icaps Summer School 2026

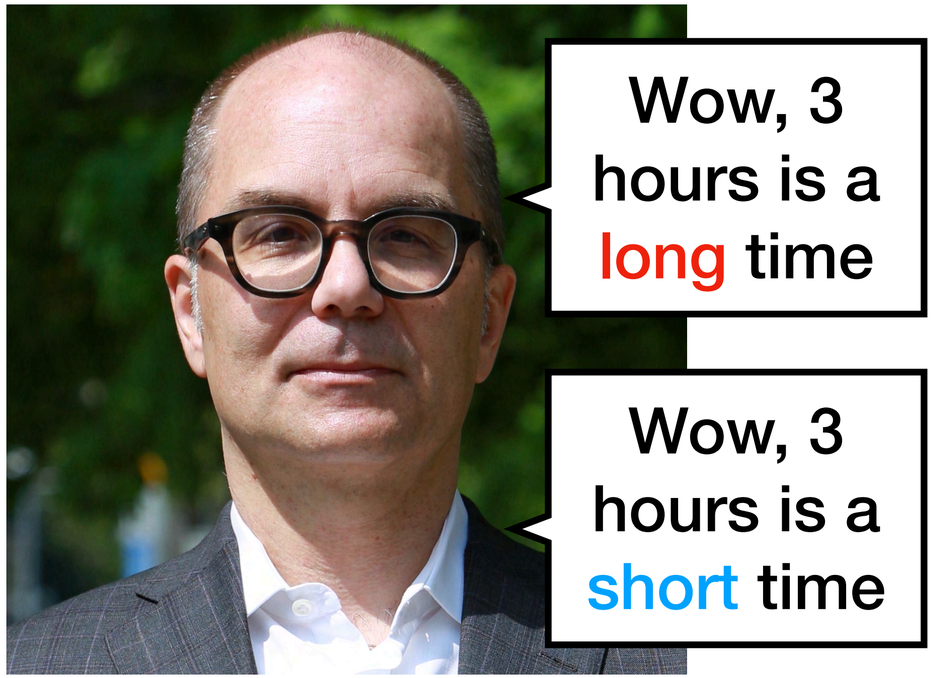
Putting the S in ICAPS

A Tutorial on Scheduling

J. Christopher Beck
Department of Mechanical & Industrial Engineering
University of Toronto
chris.beck@utoronto.ca



My First Thoughts



- I’m going to try to keep you awake
- While also introducing a little bit of scheduling research
  - Please stop me to ask questions

By Khaydock - https://commons.wikimedia.org/w/index.php?curid=25102900

The ~~Plan~~ Schedule

- Part 1: The Basics
  - What is scheduling, where is studied, computational complexity, primary solution techniques
- Part 2: Solving Three Scheduling Problems
  - Single-machine with inventory, Assembly-line balancing, Multi-project scheduling

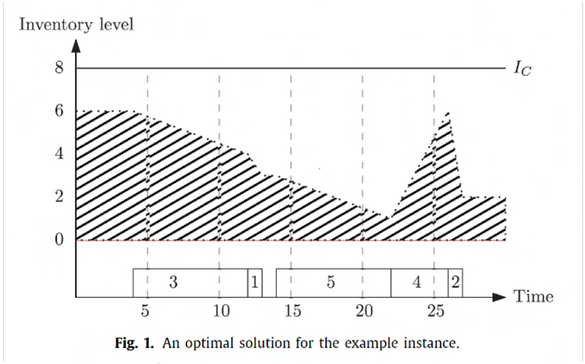
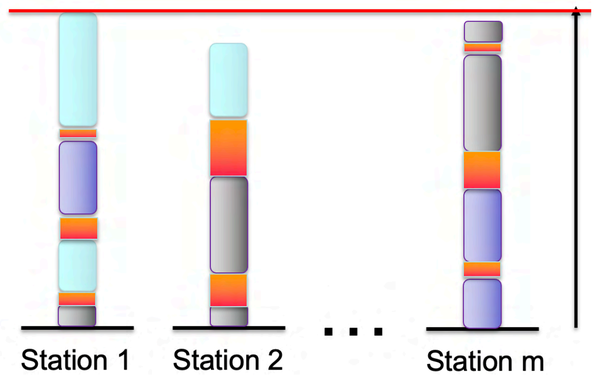
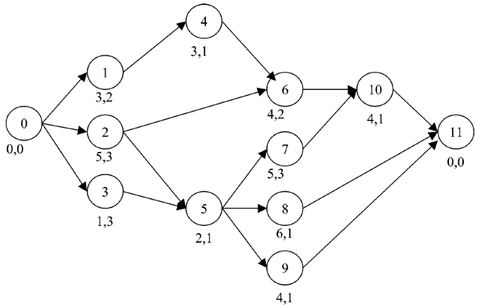
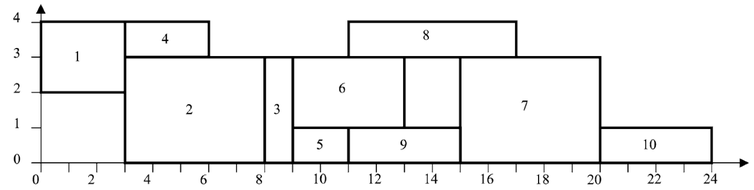


Fig. 1. An optimal solution for the example instance.







What is Scheduling?

- The allocation of scarce resources to pre-defined operations over time

machines, vehicles, fuel, people, raw materials, …

production step, work shift, time interval, …

(variable) duration, time windows, precedence relations, …

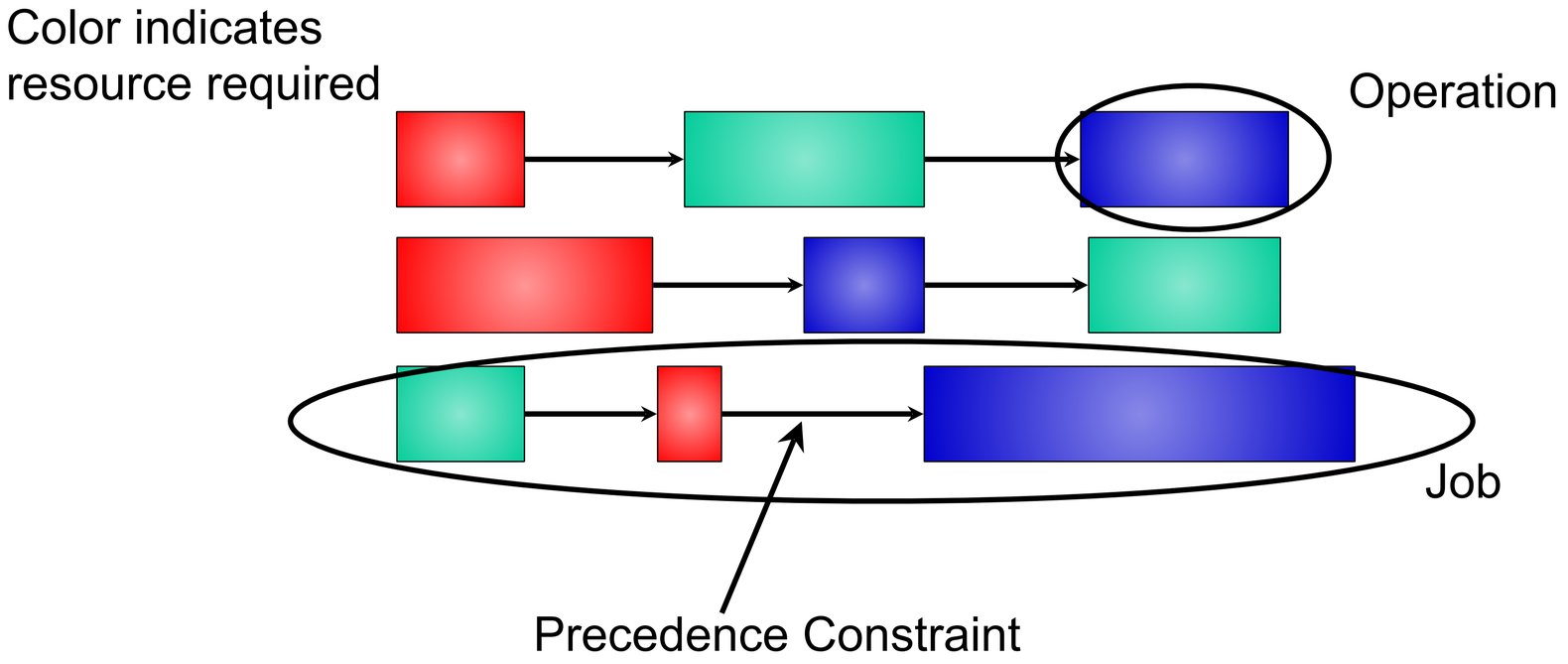
I will try to use “operation”, but may also say “task” or “activity”

- When does an operation start/end?
  What resources does it use? How much?
  How long does it take? How much does it cost?

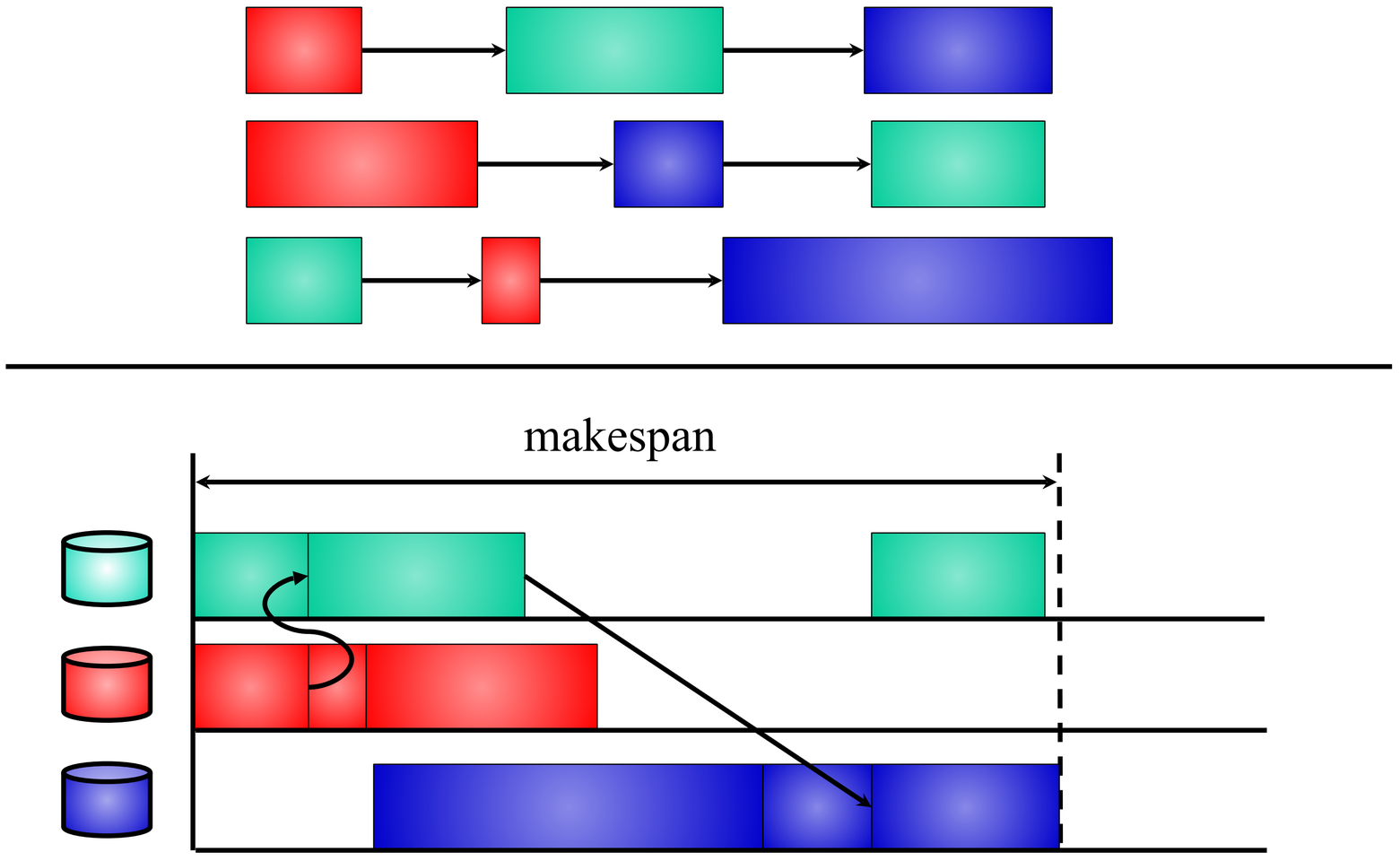
Examples

- Project scheduling 🏙️
  - build a bridge, road, sky scraper, IT application, …
- Manufacturing scheduling 🏭
  - convert and assemble final product from inputs
- Airport faculty scheduling 🛬🛫
  - allocate the runway, gate, baggage carrousel to arriving and departing planes
- Workforce scheduling 👩🏾‍⚕️☎️🍕
  - healthcare, call-centre, retail, hospitality, …

The Job Shop Scheduling Problem (JSP)



The Job Shop Scheduling Problem (JSP)



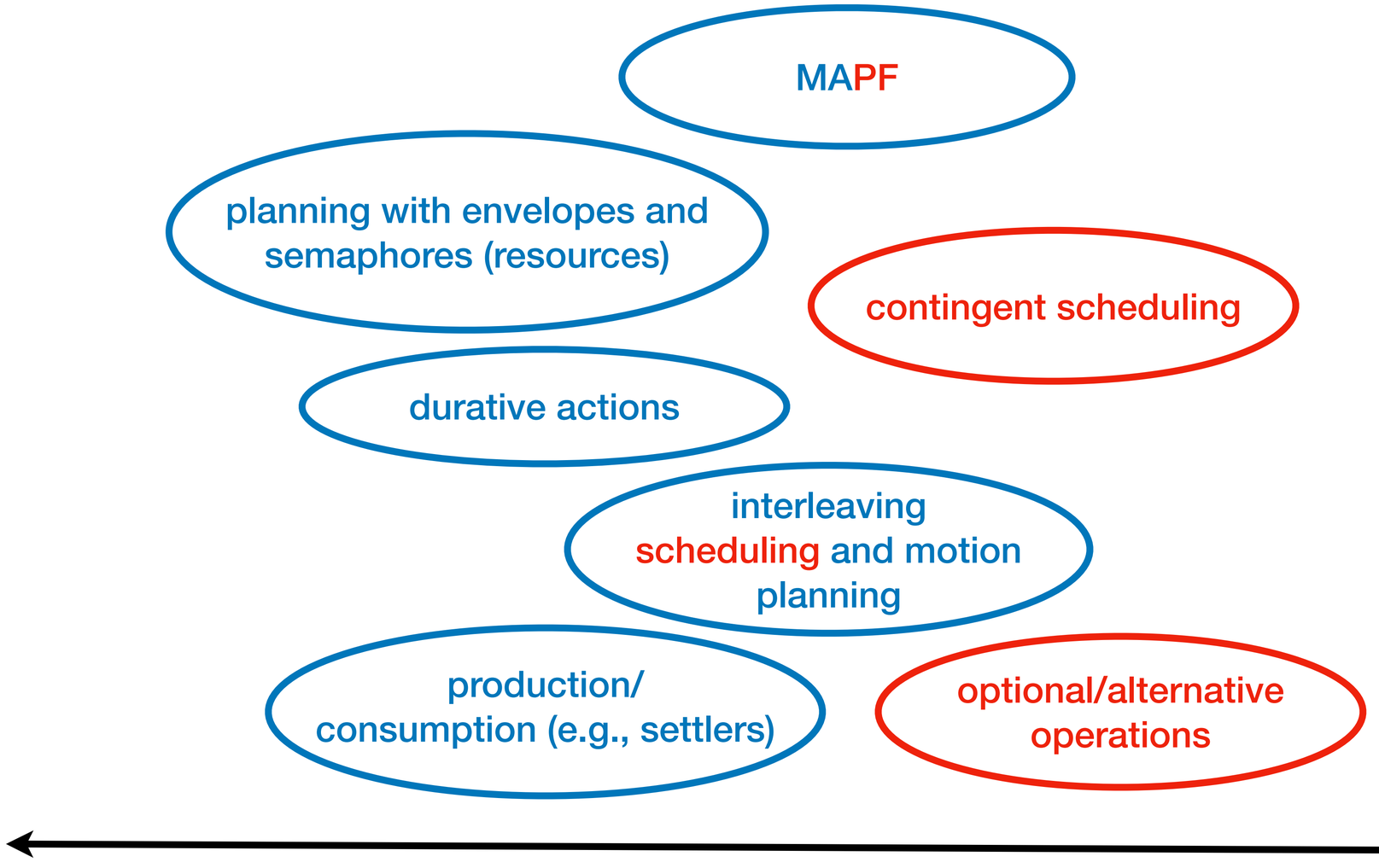
Complications

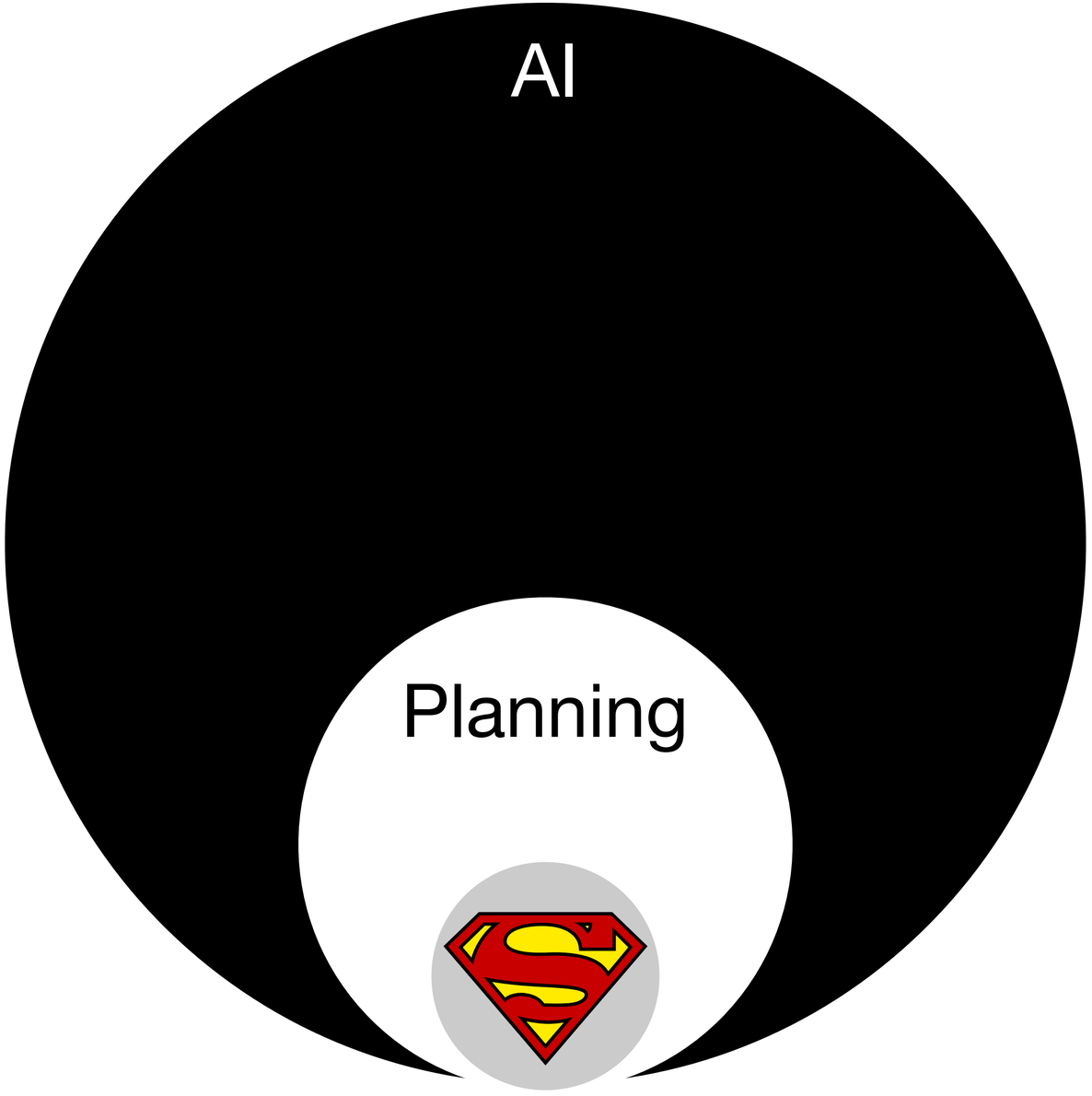
- Resources can be continuously consumed/produced
  - battery level, fuel, chemicals, …
- Batch resources require synchronization: ovens
- Sequence-dependent set-ups between operations
  - cleaning, heating, cooling, …
- Different machine speeds (“modes”)
- Multiple objective functions
  - makespan, tardiness, load balancing, quality, …
- Processing time uncertainty, arriving jobs, break downs, …

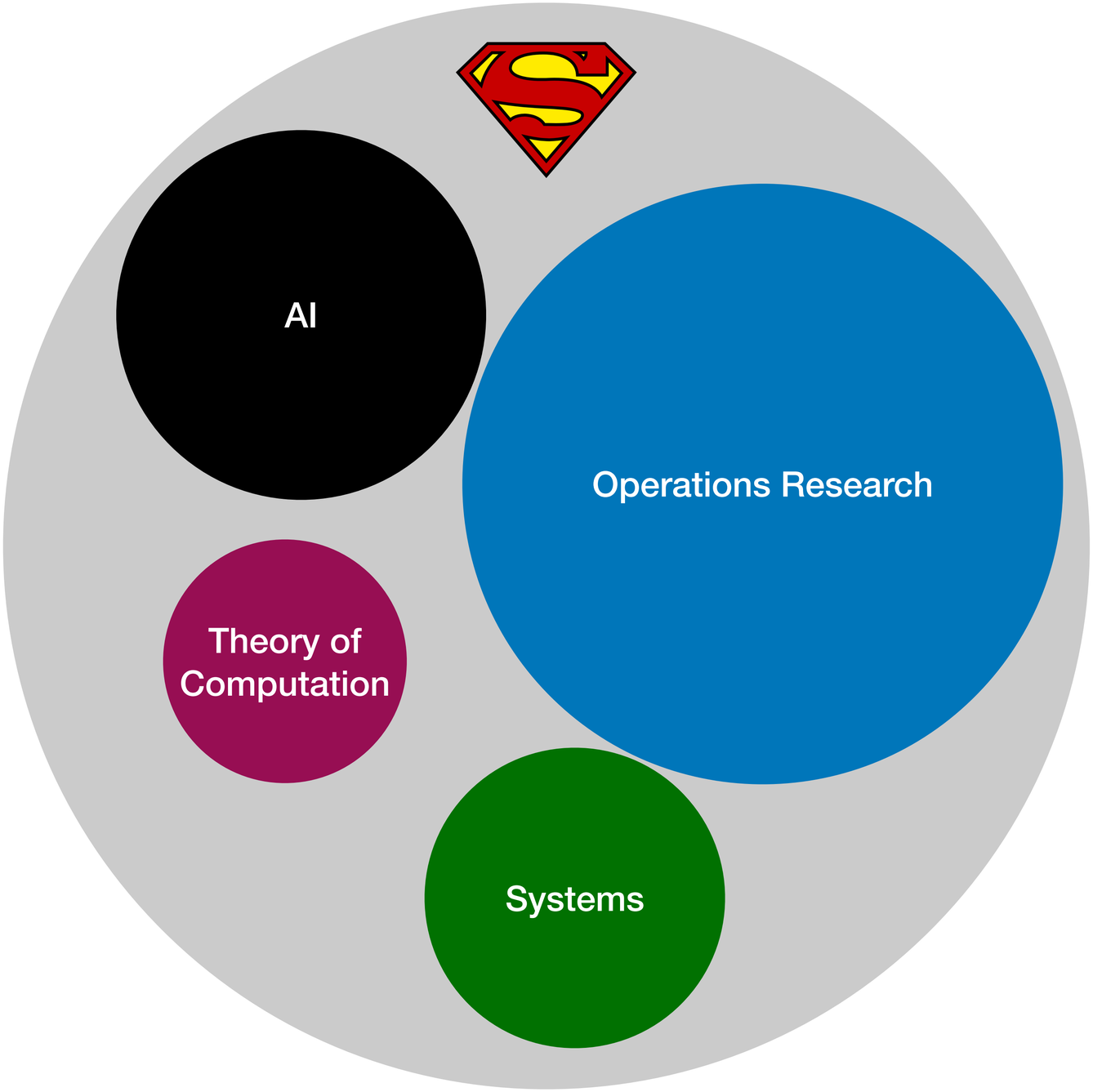
Planning vs. Scheduling

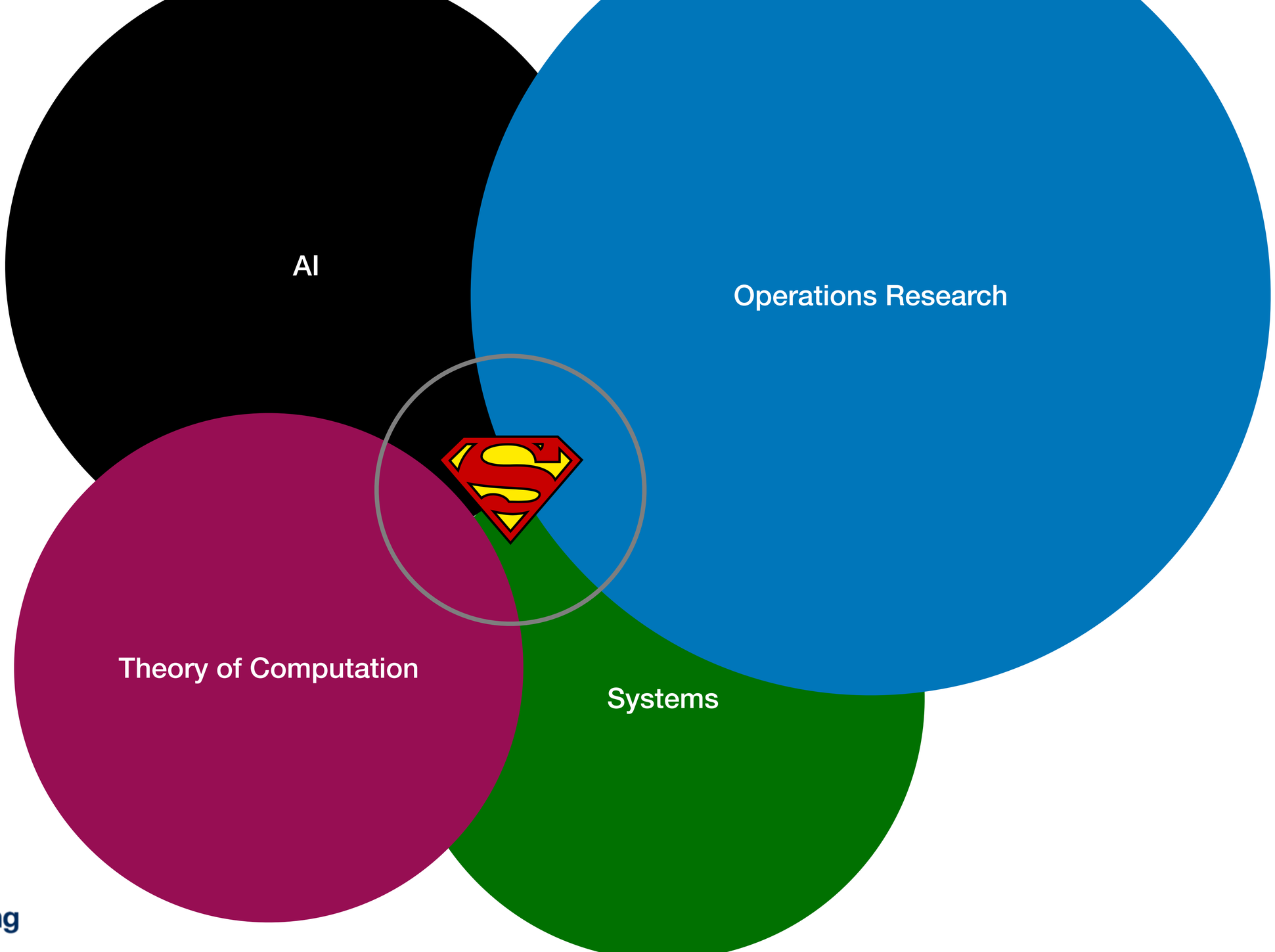
- Planning about what actions to take to achieve some goals
  - of all the things you could do, what do you do?
  - causal reasoning
- Scheduling is about when you execute a given set of operations and with what resources
  - You don’t reason about how to produce a widget, you are told how (perhaps with options)
  - You just need to decide when and with what resources

But it gets fuzzy …




AI
Planning

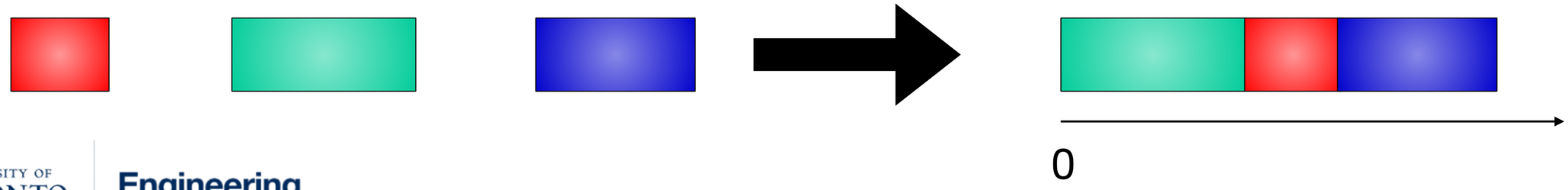

AI
Operations Research
Theory of Computation
Systems


AI
Operations Research
Theory of Computation
Systems

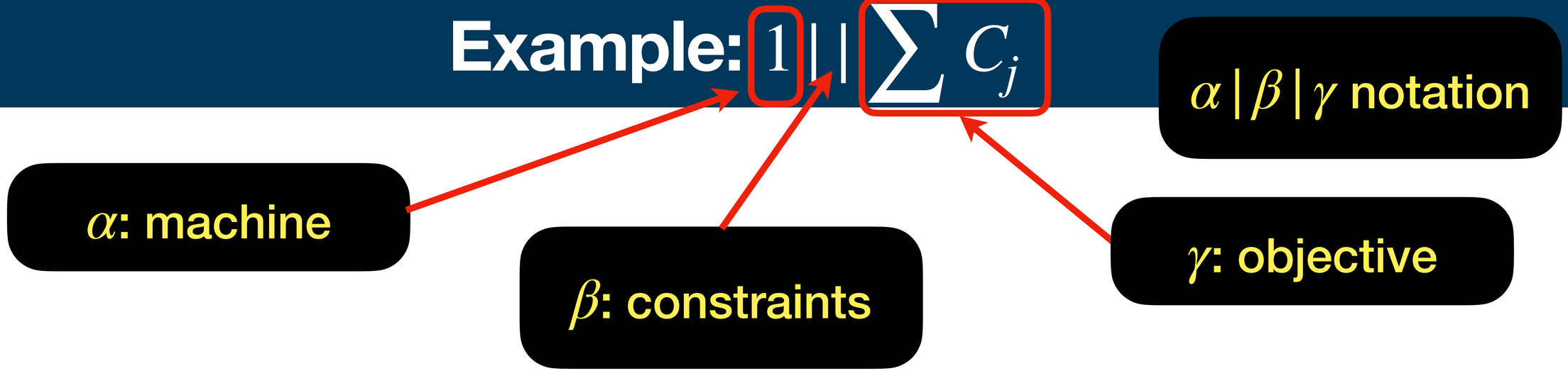
Single Machine Scheduling

*usually non-preemptively

- One machine
  - can only execute one operation at a time*
- Set of operations (“jobs”), J, each with processing time, p_j
- [Optionally] Constraints
  - e.g. release dates, partial order, …
- Minimize some objective function
  - e.g., makespan, sum of completion times, tardiness, …



Example: $1 \mid\mid \sum C_j$



- 1-machine
- no additional constraints
- minimize the sum of completion times

Computational complexity?

$1||C_{max}$

makespan

- 1-machine
- no additional constraints
- minimize maximum completion time (makespan)

Computational complexity?

Poly-time (any sequence)

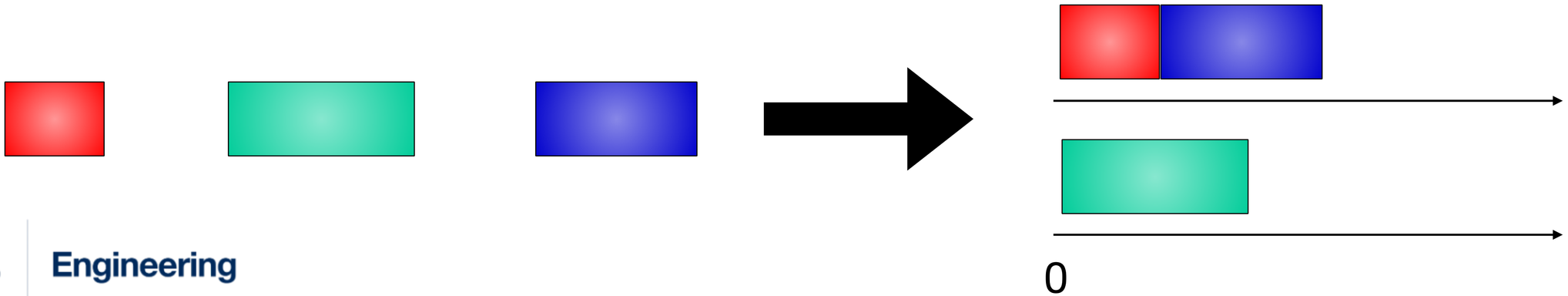
$P2||C_{max}$

"parallel"

- 2 parallel machines, you pick the one for each operation
  - operation has same processing time on both machines
- no additional constraints
- minimize the makespan

Computational complexity?

Weakly NP-hard
(number partitioning)



$1 \mid\mid \sum C_j$

sum of completion times

- 1-machine
- no additional constraints
- minimize the sum of completion times

Computational complexity?

Poly-time
(shortest processing time first (SPT))

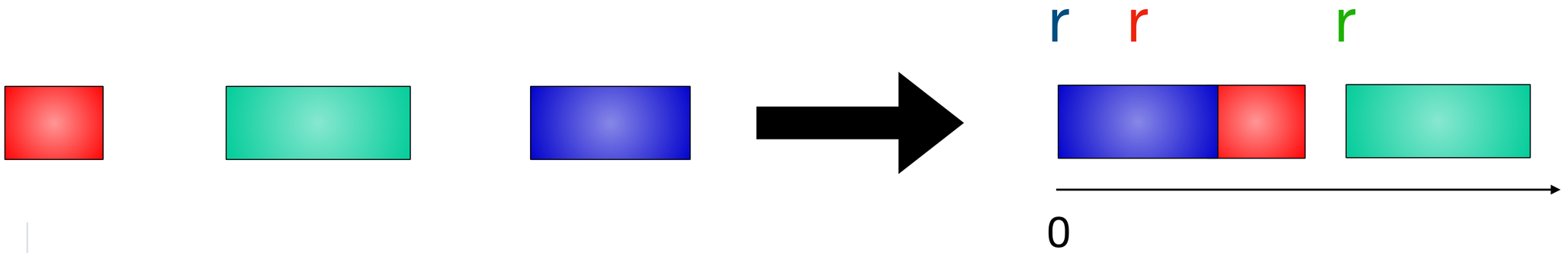
$1 \mid r_j \mid \sum C_j$

release dates

- 1-machine
- operation j must start at or after r_j
- minimize the sum of completion times

Computational complexity?

Strongly NP-hard (knapsack)



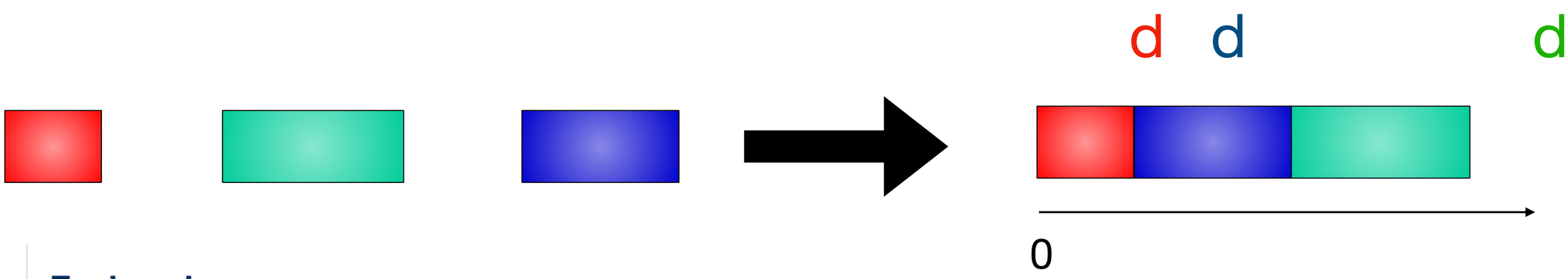
$1 \mid\mid L_{max}$

max lateness

- 1-machine
- no additional constraints
- $L_{max} = \max(C_j - d_j)$: due date: d_j

Computational complexity?

Poly-time
(earliest due-date
first (EDD))



$1 \mid\mid \sum T_j$

min tardiness

- 1-machine
- no additional constraints
- $T_j = \max(0, C_j - d_j)$: due date: d_j

Computational complexity?

NP-hard (knapsack)

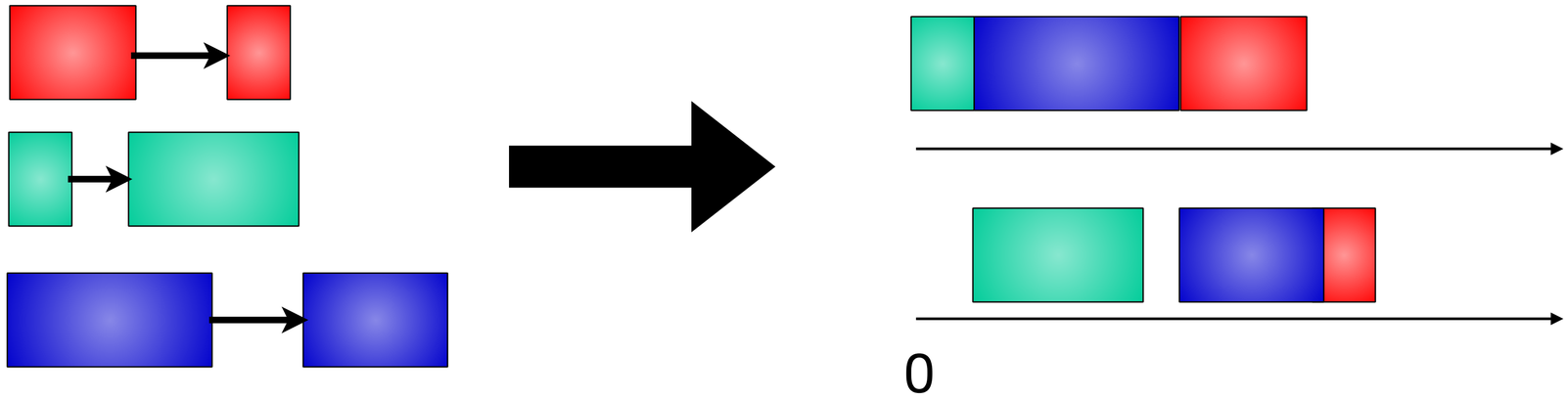
$F2 \mid\mid C_{max}$

"flowshop"

Computational complexity?

- 2-machine flowshop
  - a "job" is 2 ordered operations
  - first operation on M1, second on M2
  - no additional constraints
- minimize makespan

Poly-time
(Johnson's Rule)



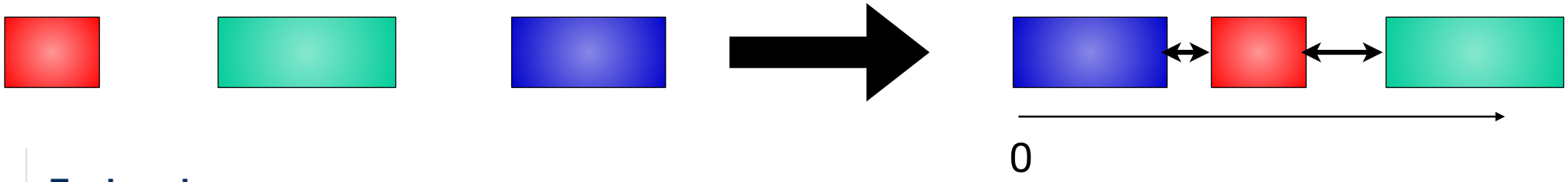
$1 \mid s_{ij} \mid C_{max}$

sequence dependent set-ups

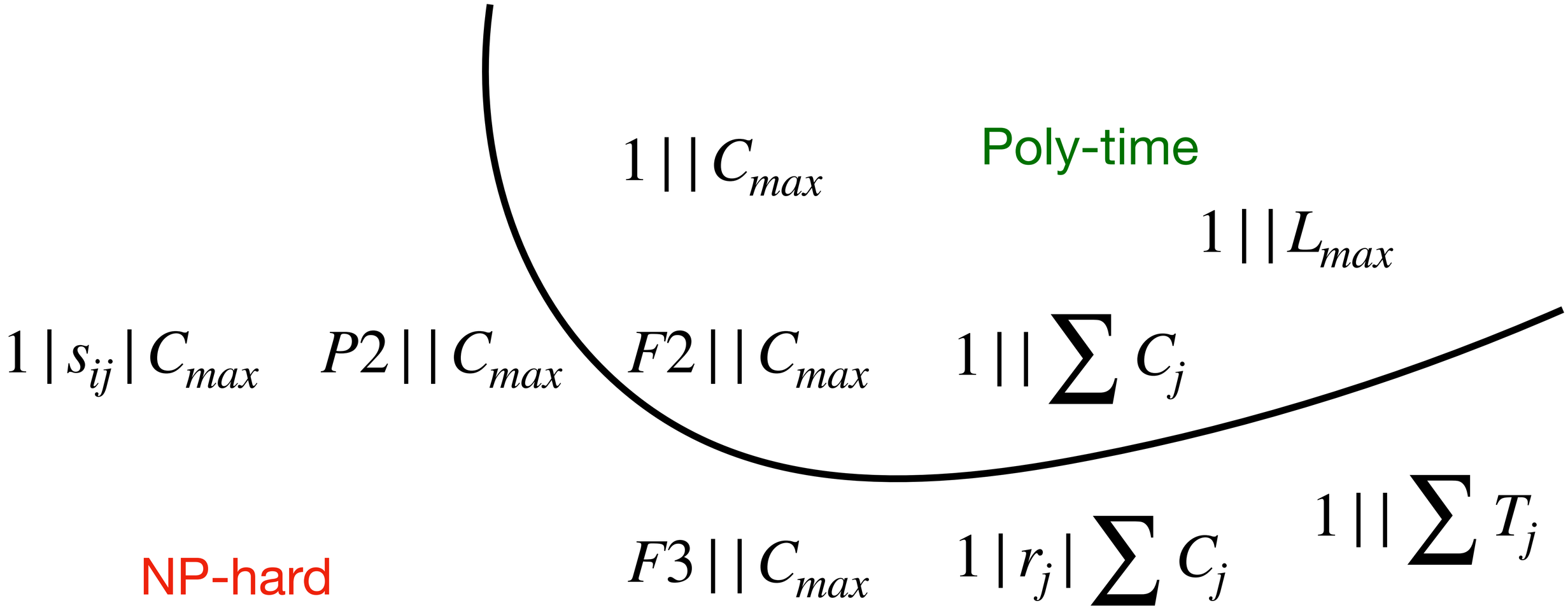
- 1-machine
- "set-up" time of at least s_{ij} between end of i and start of j
- minimize makespan

Computational complexity?

Strong NP-hard (TSP)



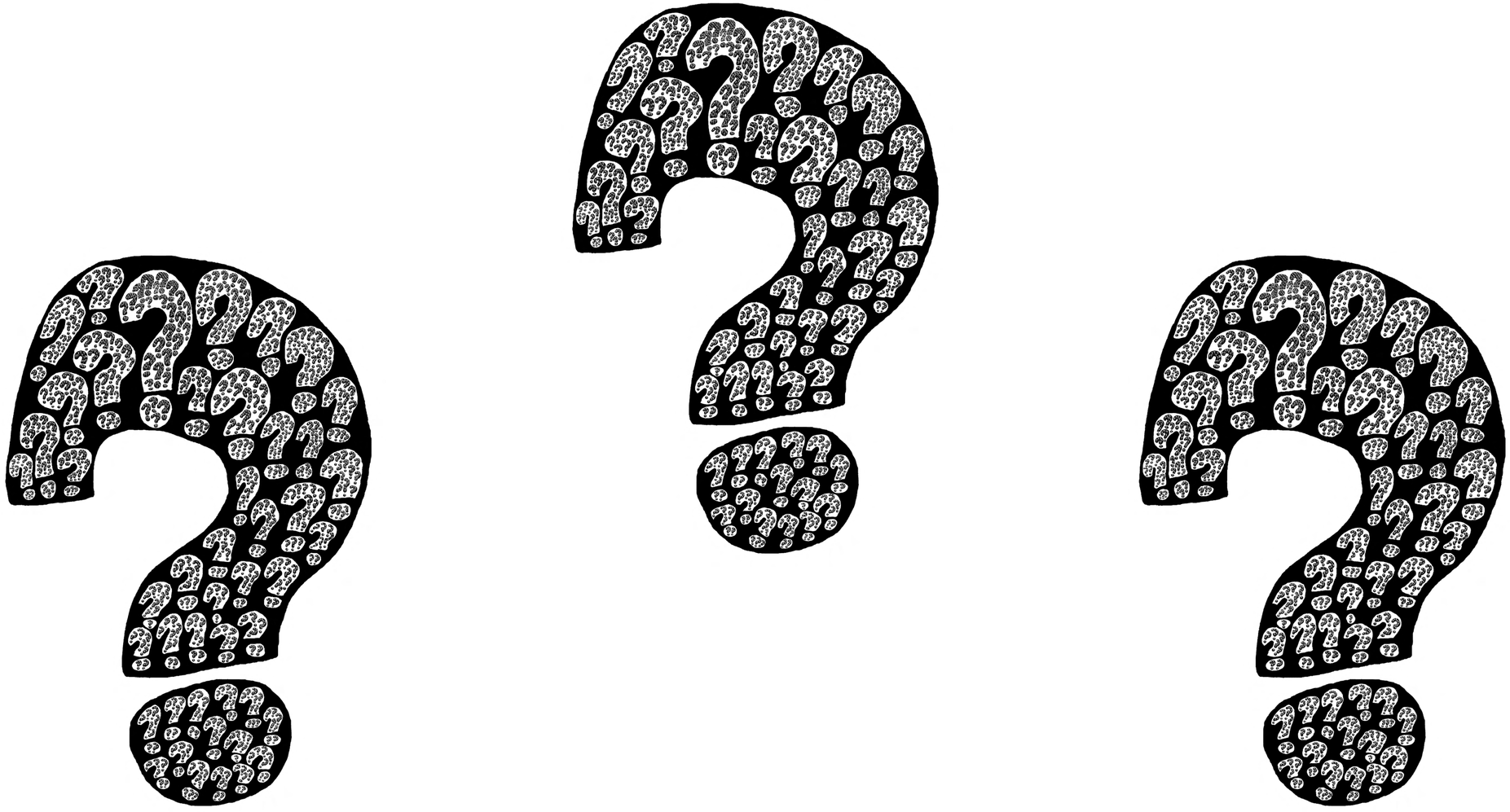
A Very Partial Map



An Important Question

- People who like Theory of Computation
  - Interesting and non-obvious easy/ hard border
- People who want to solve scheduling problems
  - If a relaxation gives us a poly-time problem, we can exploit it

Questions?



By Khaydock - https://commons.wikimedia.org/w/index.php?curid=25102900

Solving Scheduling Problems

- Dispatch Rules
  - Optimal rules to sequence operations for easy problems: SPT, EDD, Johnson's rule, …
- Customized Branch-and-Bound and Dynamic Programming algorithms
- All the meta-heuristics
- Mathematical Programming
  - Mixed Integer Linear Programming, Constraint Programming
- And why not heuristic search?

Mixed Integer Linear Programming (MILP or MIP)

What is MILP?

- Very successful, sound and complete optimization technique
- Origins in the 1940s and 1950s in Operations Research and Applied Math
- Commercial solvers: Gurobi, IBM CPLEX, FICO Xpress, Cardinal, …
- Widely taught in business schools and used industrially for all sorts of optimization problems
- Strong mathematical and computational foundations

What's a MILP?

Can express NP-hard problems

$$\min \quad \sum_{i \in V} c_i x_i$$

Linear objective function

$$\sum_{i \in V} a_{ij} x_i \leq b_j \qquad \forall j \in C$$

Linear constraints

$$V = V_I \cup V_R$$

$$x_i \in \mathbb{Z} \qquad \forall i \in V_I$$

Integer variables

$$x_i \in \mathbb{R} \qquad \forall i \in V_R$$

Continuous variables

Continuous (linear) relaxation: poly-time soluble!

$x_i \in \mathbb{Z}$ is Poly-time Solvable

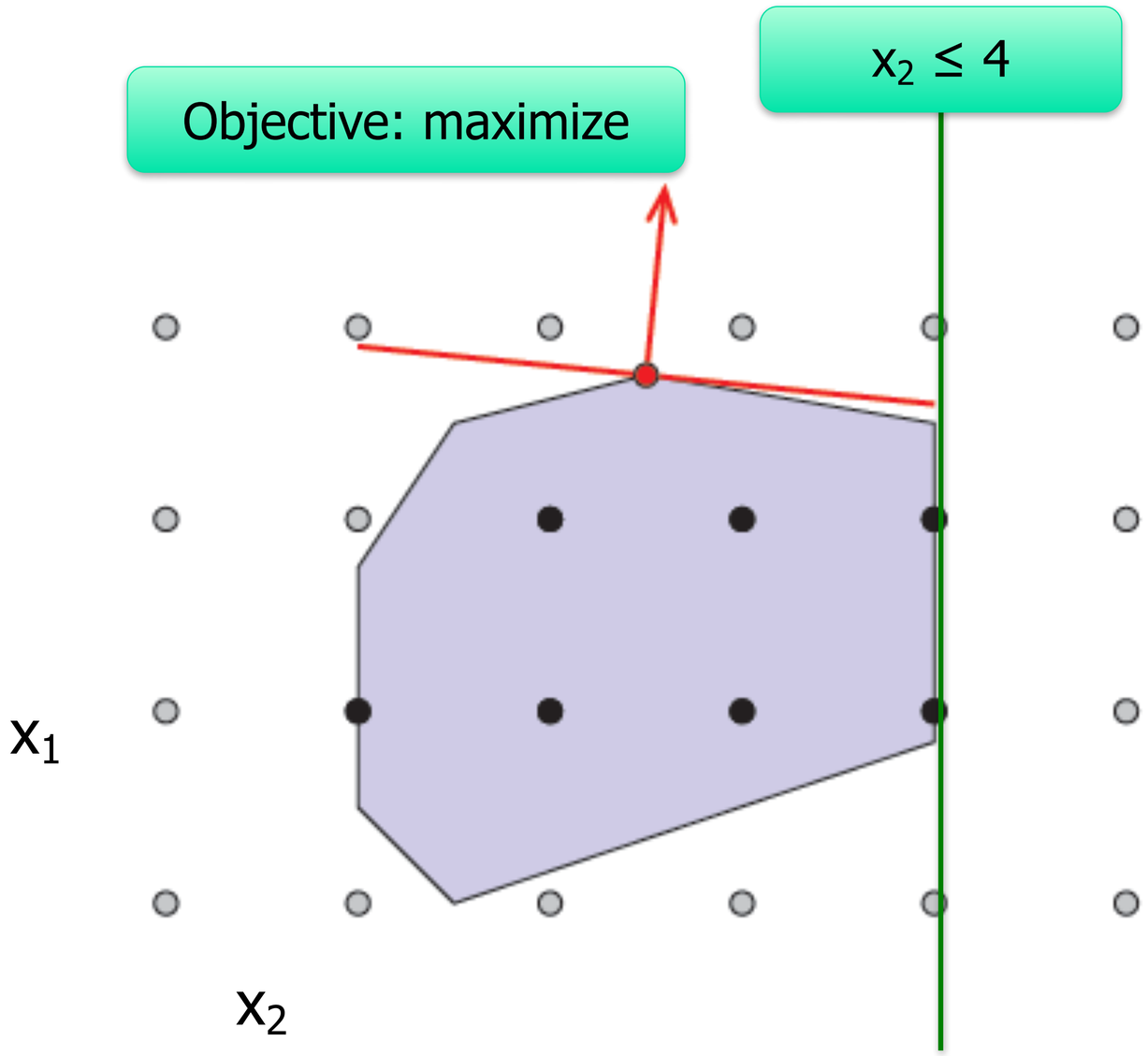
- Almost everything we know about MILP solving stems from the fact that relaxing the integrality constraints results in a linear program
  - The foundational math: linear algebra, polyhedral analysis
  - The solving approaches depend on solving or tightening or transforming or understanding the linear relaxation

If you are looking for a beautiful and useful area of applied math to get into (you know, in your spare time), this is a good one

Solving MILPs

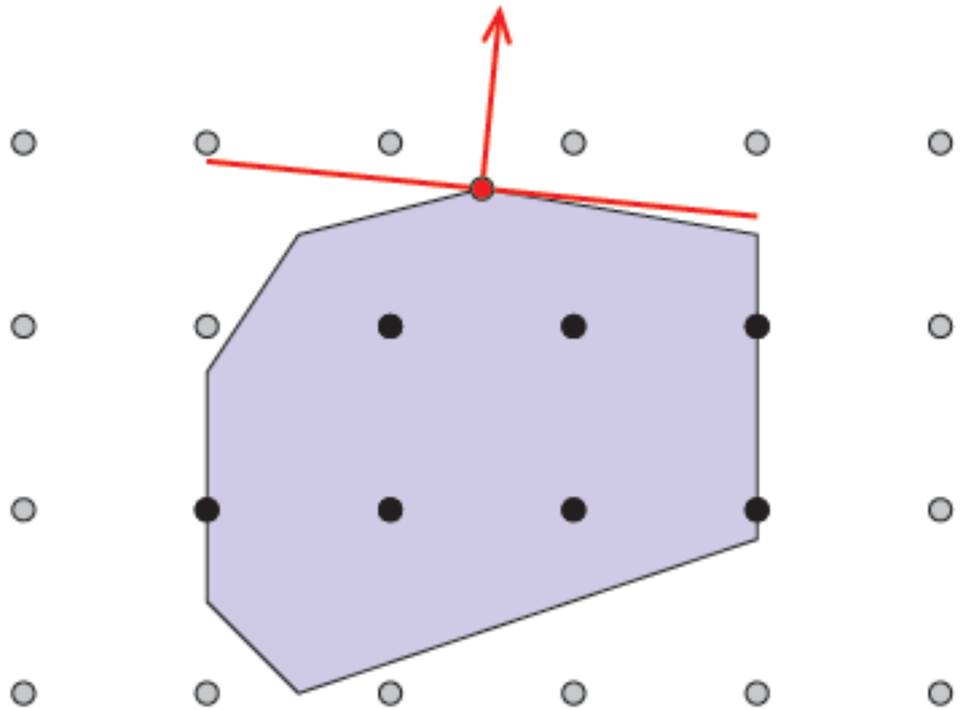
- "Branch-bound-and-cut"
  - tree search
  - solving the LP relaxation
  - adding cutting planes

There is a mistake coming up in a few slides. See if you can spot it.



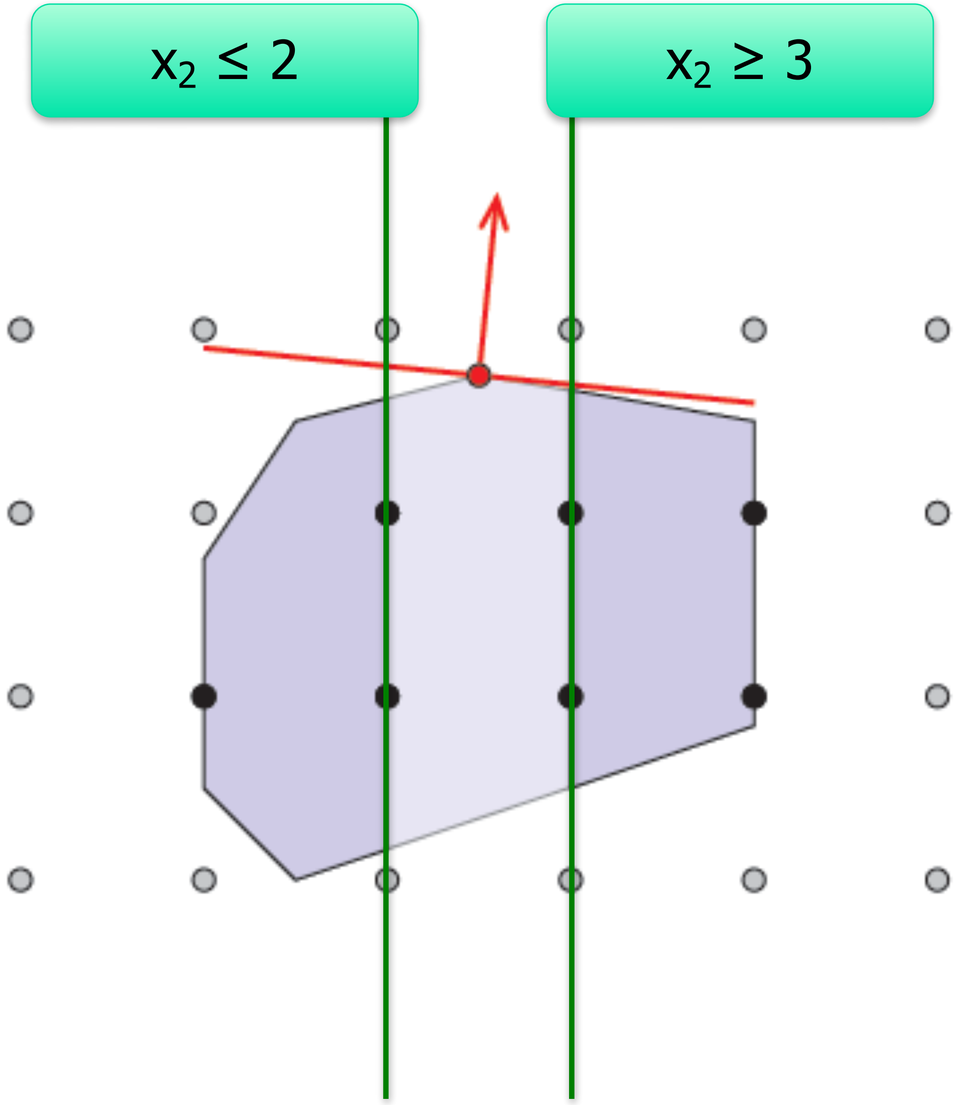
Solving the Linear Relaxation

- Finds a “corner” that maximizes the objective function
  - simplex, dual simplex, interior point, …



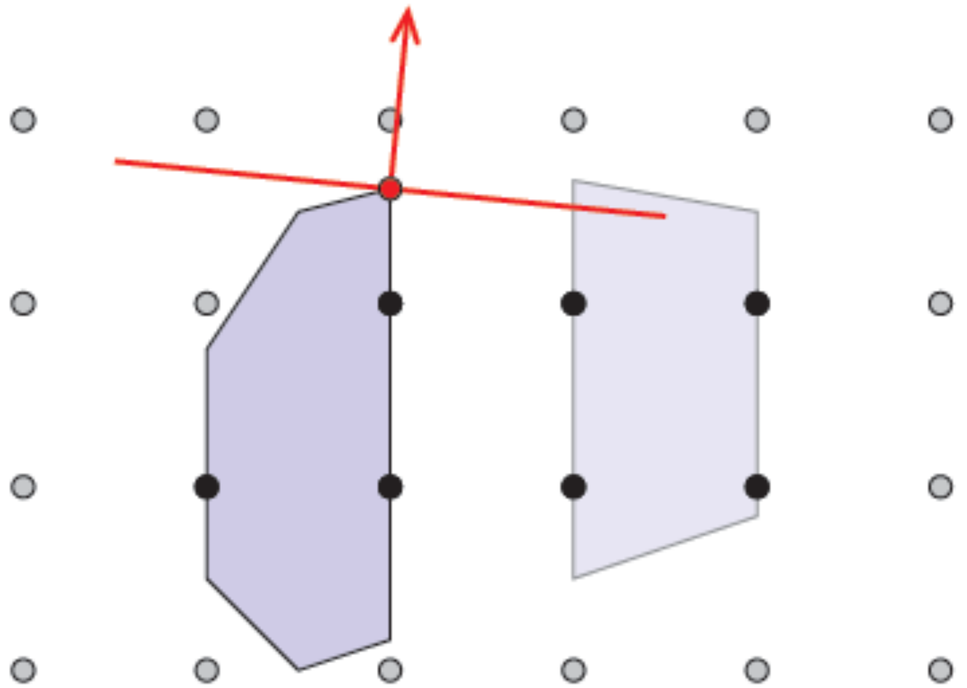
Branching

- Add a constraint to partition the feasible space
- Solve one partition
- Return later to the other partition (backtrack)



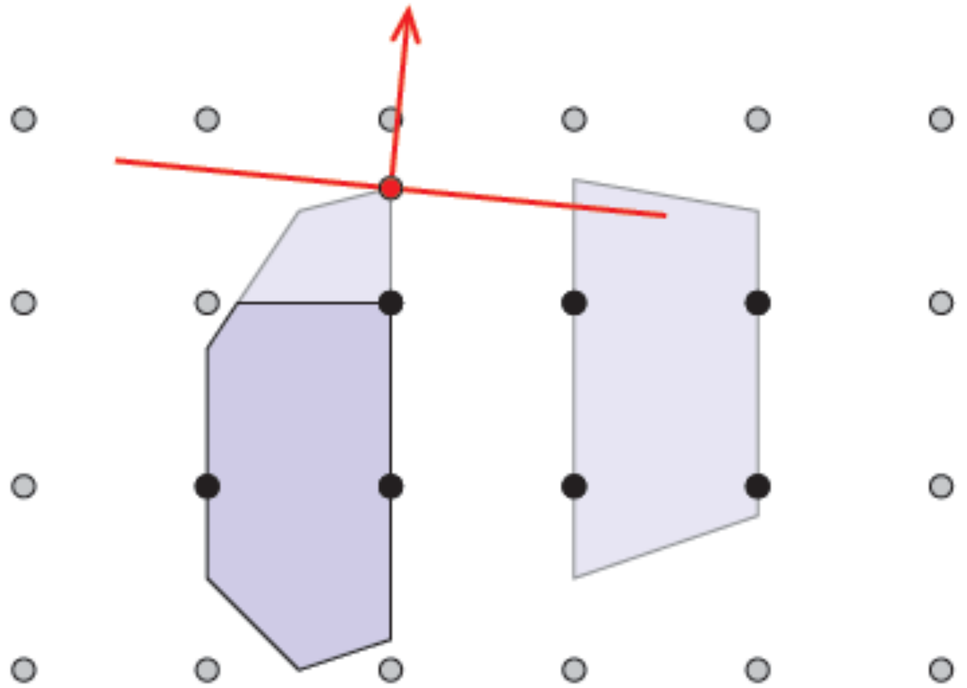
Branching

- Add a constraint to partition the feasible space
- Solve one partition
- Return later to the other partition (backtrack)



Bounds Strengthening

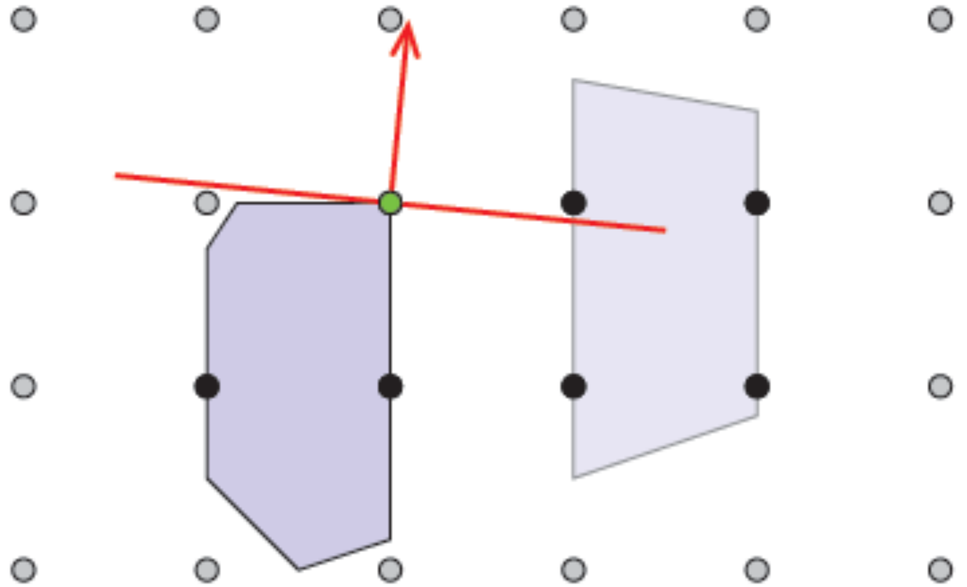
- A form of inference



Solve LP Again

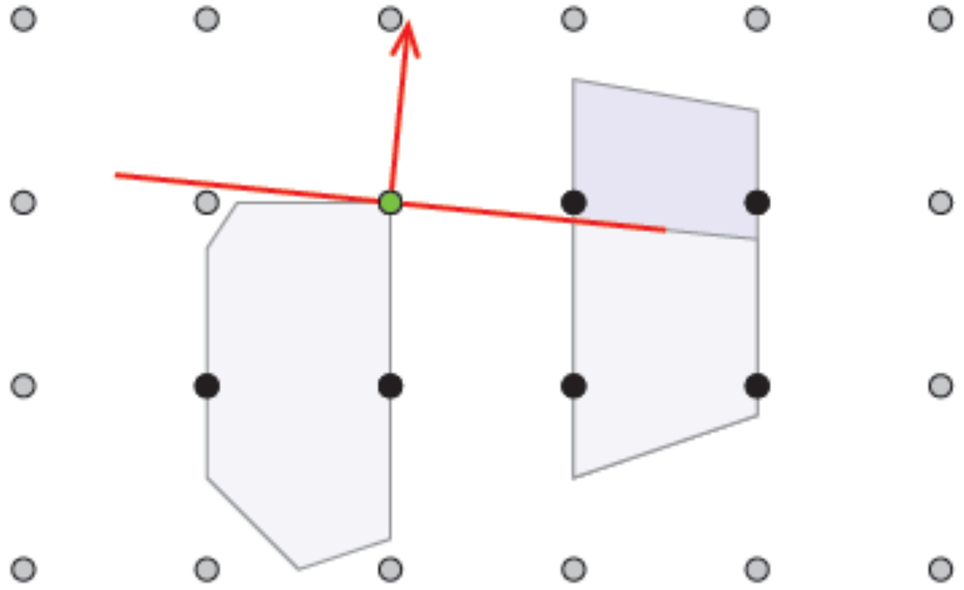
- Integer solution!

So are we done?

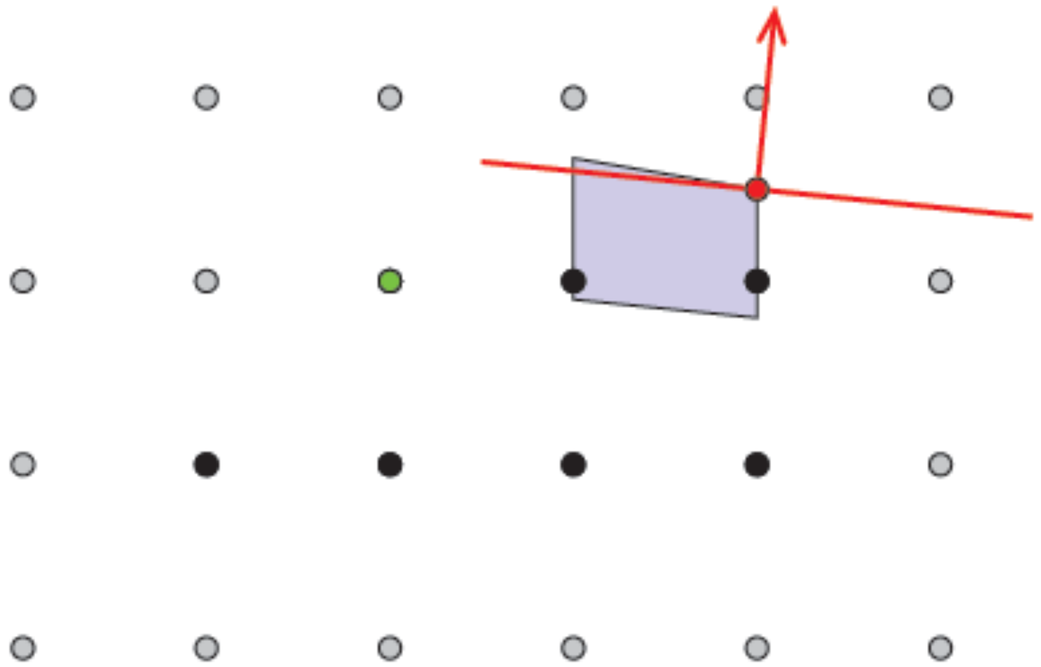


Bounding

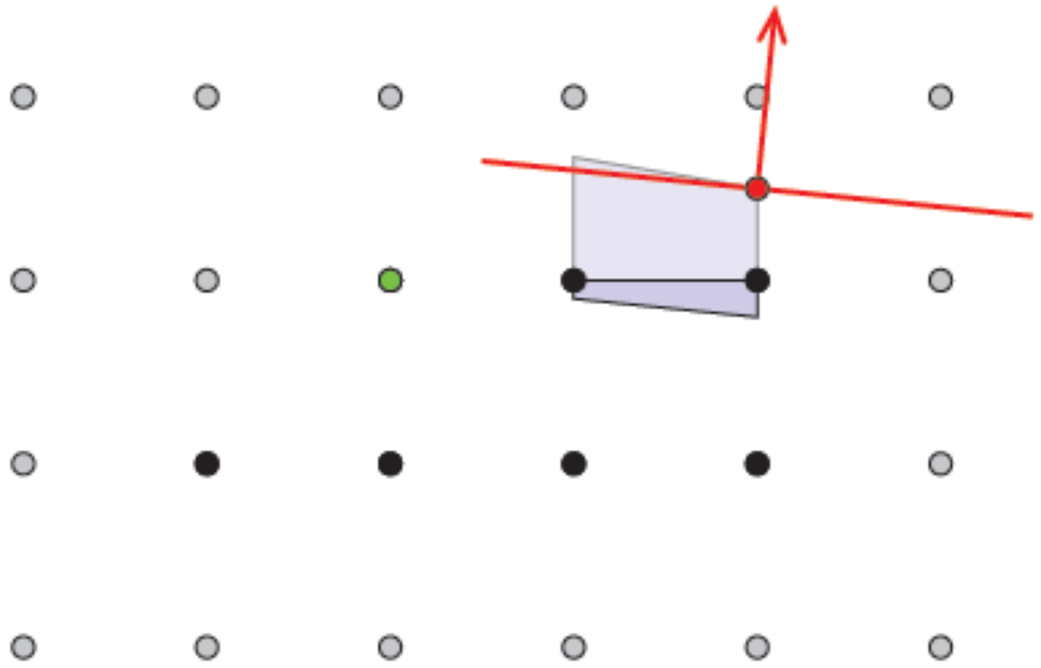
- We found an incumbent solution
  - add a new constraint so that we only search for a better one



Backtrack

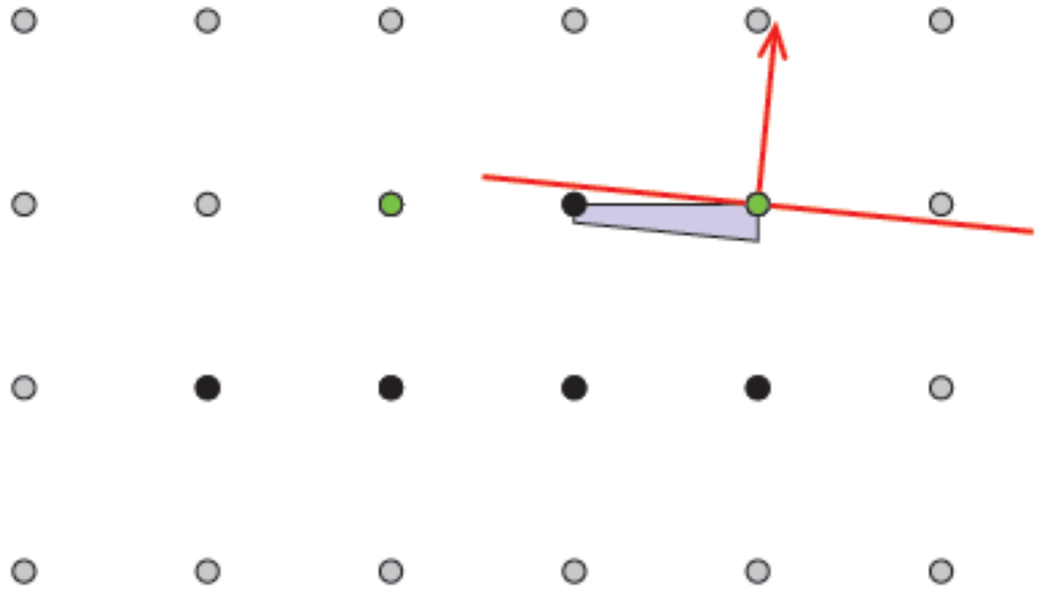


Strengthen Bounds



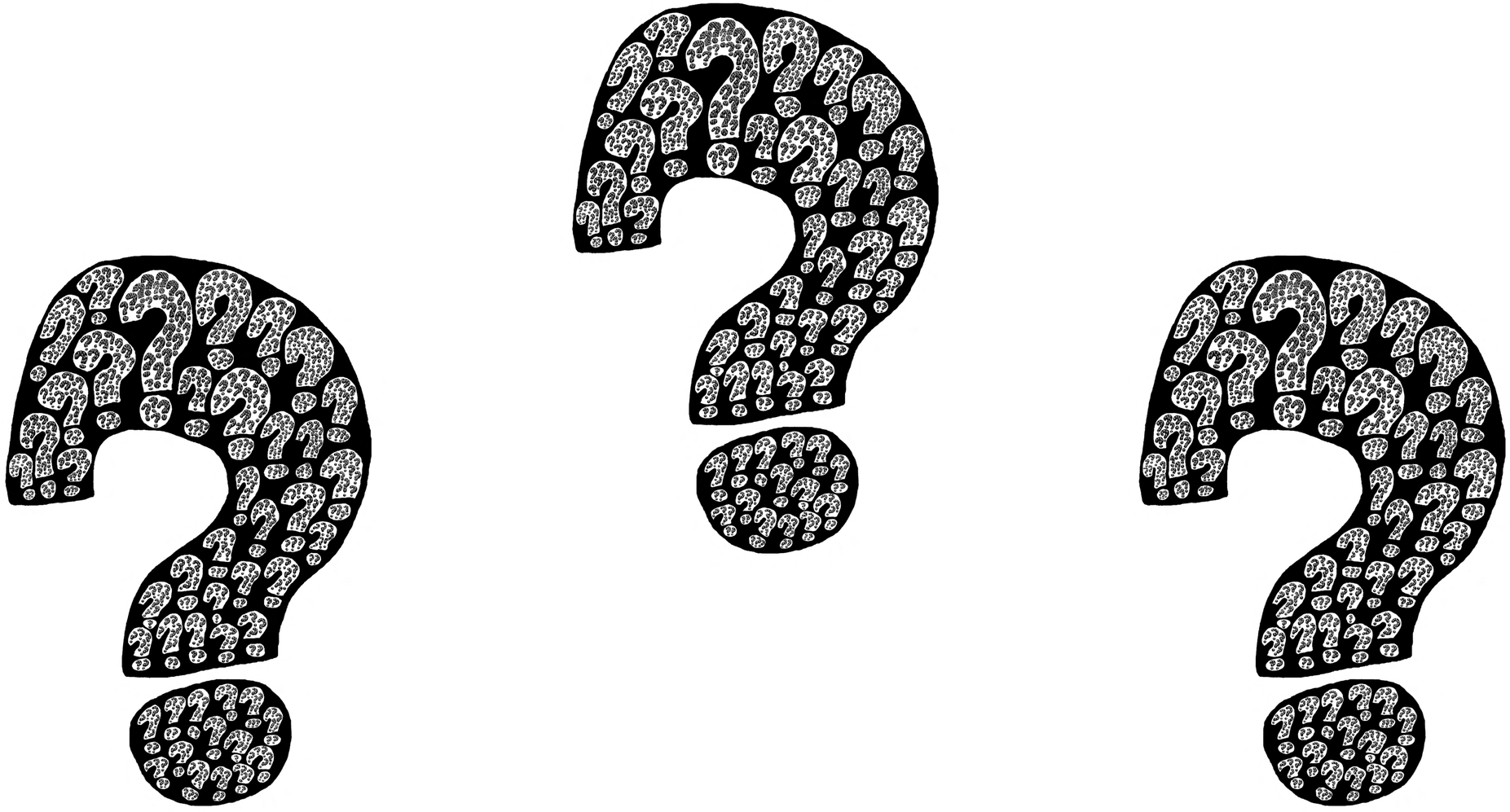
New Integer Solution

- And we're done



Did you see the mistake?

Questions?



By Khaydock - https://commons.wikimedia.org/w/index.php?curid=25102900

Modeling in MILP

$$\min \quad \sum_{i \in V} c_i x_i$$

Linear objective function

$$\sum_{i \in V} a_{ij} x_i \leq b_j \qquad \forall j \in C$$

Linear constraints

$$V = V_I \cup V_R$$

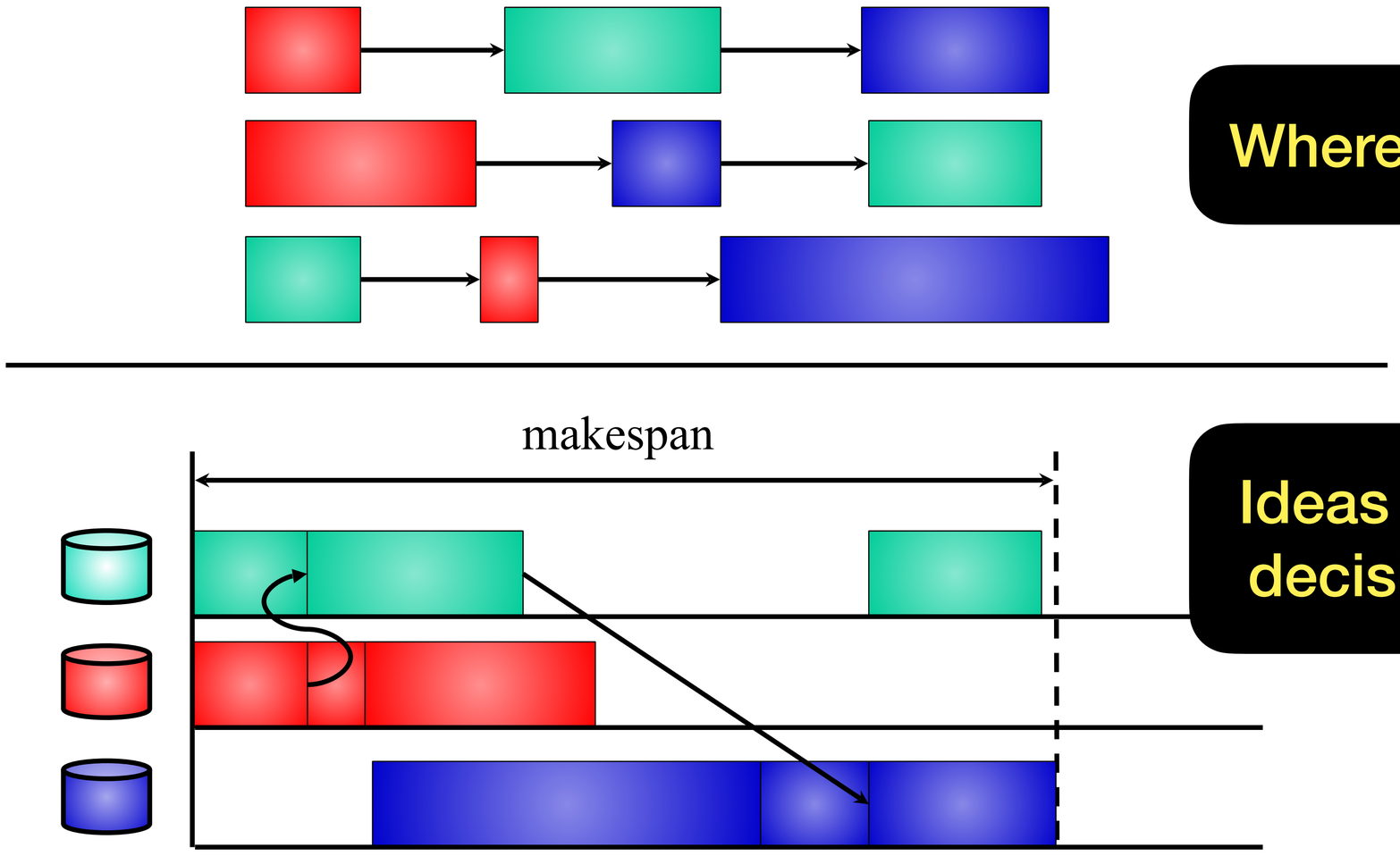
$$x_i \in \mathbb{Z} \qquad \forall i \in V_I$$

Integer variables

$$x_i \in \mathbb{R} \qquad \forall i \in V_R$$

Continuous variables

The Job Shop Scheduling Problem (JSP)



Where do we start?

Ideas for a primary decision variable?

Time-Indexed JSP MILP Model

- The main decision variable represents if an operation j starts at time ($= 1$) t or not ($= 0$)
  - Variables are “indexed” by time

Notation

- $\mathcal{H}$ is the set of all time-points
- $T_{jt} = \{t - p_j + 1, \ldots, t\}$ Set of time points, defined by j and t
- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
- $x_{jt} = 1$ iff operation j starts at time t

What Constraints Do We Need?

- Each operation starts exactly once
- Makespan is the maximum completion time
- Precedence constraints respected
- No machine is used by more than one operation at a time

– $\mathcal{H}$ is the set of all time-points
– $T_{jt} = \{t - p_j + 1, \ldots, t\}$
– $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
– $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
– $x_{jt} = 1$ iff operation j starts at time t

Time-Indexed JSP MILP Model

$$\min \quad C_{max}$$

$$\text{s.t.} \quad \sum_{t \in \mathcal{H}} x_{jt} = 1 \qquad \forall j \in \mathcal{J}$$

All operations start only once

$$\sum_{j \in \mathcal{J}_k} \sum_{t' \in T_{jt}} x_{jt'} \leq 1 \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H}$$

Resource constraints

$$\sum_{t \in \mathcal{H}} (t + p_j) x_{jt} \leq C_{max} \qquad \forall j \in \mathcal{J}$$

C_{max} is the largest end-time

$$\sum_{t \in \mathcal{H}} (t + p_j) x_{jt} \leq \sum_{t \in \mathcal{H}} (t) x_{it} \qquad \forall (j, i) \in \mathcal{E}$$

Precedence constraints

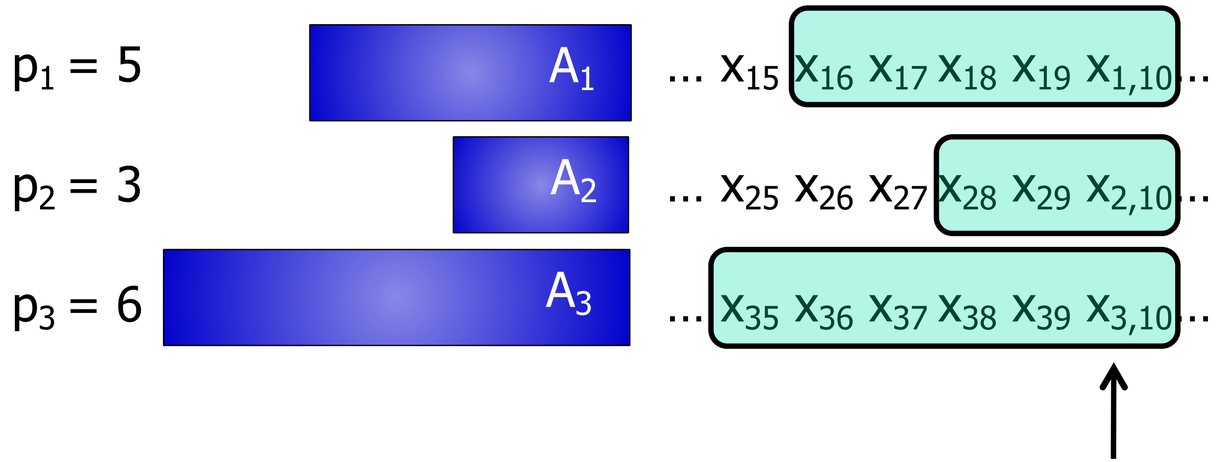
$$x_{jt} \in \{0, 1\} \qquad \forall j \in \mathcal{J}, \ \forall t \in \mathcal{H}$$

- $\mathcal{H}$ is the set of all time-points
- $T_{jt} = \{t - p_j + 1, \ldots, t\}$
- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
- $x_{jt} = 1$ iff operation j starts at time t

Resource Constraints

$$\sum_{j \in \mathcal{J}_k} \sum_{t' \in T_{jt}} x_{jt'} \leq 1 \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H}$$

$$T_{jt} = \{t - p_j + 1, \ldots, t\}$$



Make sure that the blue resource is not over-capacitated at time 10

Time-Indexed JSP MILP Model

$$\min \quad C_{max}$$

$$\text{s.t.} \quad \sum_{t \in \mathcal{H}} x_{jt} = 1 \qquad \forall j \in \mathcal{J}$$

All operations start only once

$$\sum_{j \in \mathcal{J}_k} \sum_{t' \in T_{jt}} x_{jt'} \leq 1 \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H}$$

Resource constraints

$$\sum_{t \in \mathcal{H}} (t + p_j) x_{jt} \leq C_{max} \qquad \forall j \in \mathcal{J}$$

C_{max} is the largest end-time

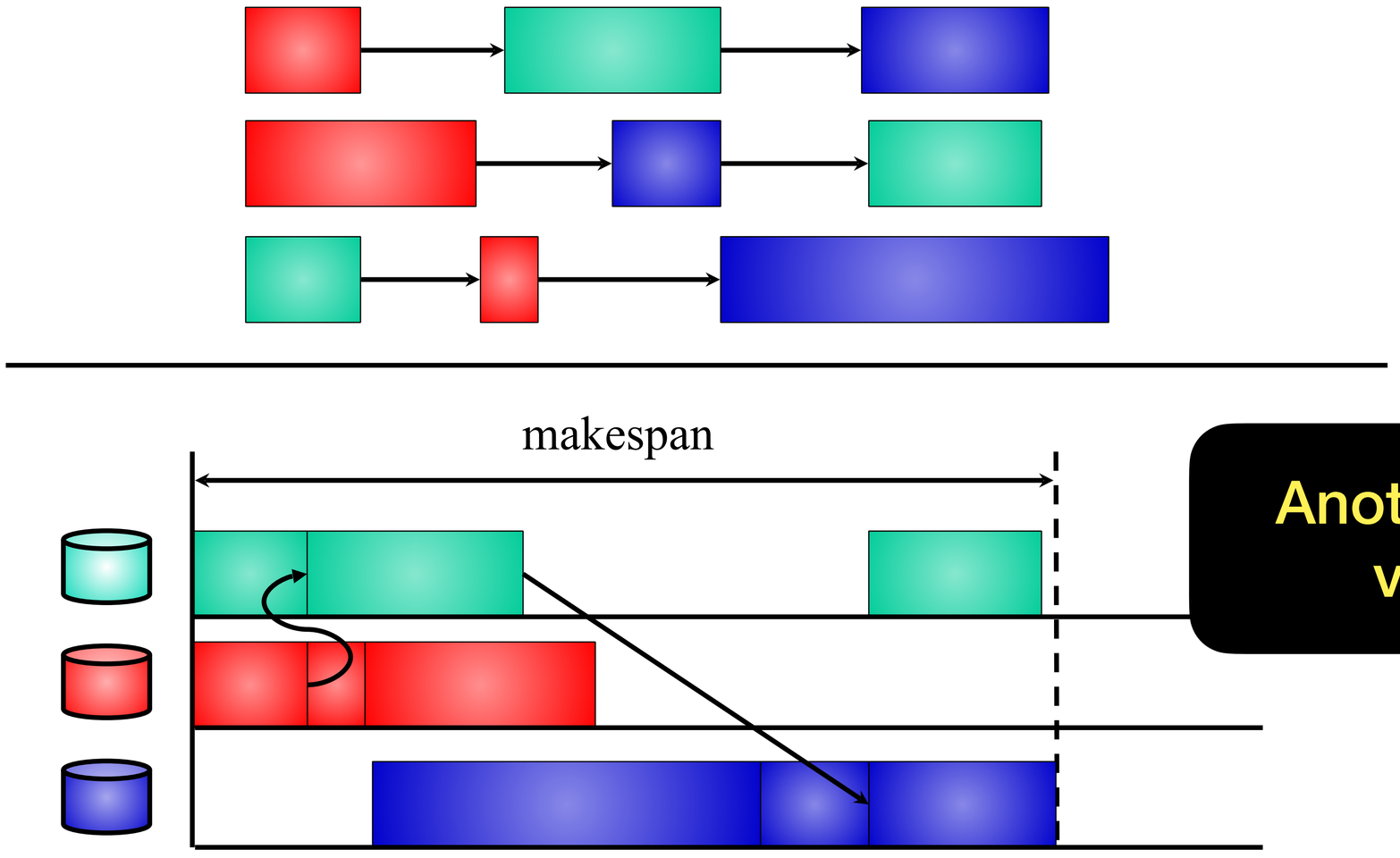
$$\sum_{t \in \mathcal{H}} (t + p_j) x_{jt} \leq \sum_{t \in \mathcal{H}} (t) x_{it} \qquad \forall (j, i) \in \mathcal{E}$$

Precedence constraints

$$x_{jt} \in \{0, 1\} \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{H}$$

- $\mathcal{H}$ is the set of all time-points
- $T_{jt} = \{t - p_j + 1, \ldots, t\}$
- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
- $x_{jt} = 1$ iff operation j starts at time t

The Job Shop Scheduling Problem (JSP)



Another primary variable?

Disjunctive JSP MILP Model

- The main decision variable represents the sequence of a pair of operations on the same machine
  - i before j or j before i

Notation

- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
- M a big constant
- S_j is the start-time of operation j
- $z_{ji} = 1$ iff operation i is scheduled after operation j

What Constraints Do We Need?

- Each operation starts exactly once
- Makespan is the maximum completion time
- Precedence constraints respected
- No machine is used by more than one operation at a time

Where:

- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j, i) s.t. i is constrained to be after j
- M a big constant
- S_j is the start-time of operation j
- $z_{ji} = 1$ iff operation i is scheduled after operation j

Disjunctive JSP MILP Model

$$
\begin{aligned}
\min \quad & C_{max} \\
\text{s.t.} \quad & S_j + p_j \leq C_{max} && \forall j \in \mathcal{J} \\
& S_j + p_j \leq S_i && \forall (j,i) \in \mathcal{E} \\
& S_j \geq S_i + p_i - M \cdot z_{ji} && j,i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K} \\
& S_i \geq S_j + p_j - M \cdot (1 - z_{ji}) && j,i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K} \\
& 0 \leq S_j \leq |\mathcal{H}| && \forall j \in \mathcal{J} \\
& z_{ji} \in \{0,1\} && \forall j,i \in \mathcal{J}_k, \forall k \in \mathcal{K}
\end{aligned}
$$

C_{max} is the largest end-time

Precedence constraints

Resource constraints

Where:

- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j,i) s.t. i is constrained to be after j
- M a big constant
- S_j is the start-time of operation j
- $z_{ji} = 1$ iff operation i is scheduled after operation j

Resource Constraints

$$S_j \geq S_i + p_i - M \cdot z_{ji} \qquad j, i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K}$$
$$S_i \geq S_j + p_j - M \cdot (1 - z_{ji}) \qquad j, i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K}$$

- $z_{ji} = 1 \rightarrow i$ after j
  - $S_j \geq S_i + p_i - M$: redundant since $S_j \geq 0$
  - $S_i \geq S_j + p_j - 0$: precedence constraint
- $z_{ji} = 0 \rightarrow j$ after i
  - $S_j \geq S_i + p_i - 0$: precedence constraint
  - $S_i \geq S_j + p_j - M$: redundant since $S_i \geq 0$

z_{ji} is a switch between 2 constraints

"big-M" constraint is a common modeling MILP modeling pattern

Disjunctive JSP MILP Model

$$
\begin{aligned}
\min \quad & C_{max} \\
\text{s.t.} \quad & S_j + p_j \leq C_{max} && \forall j \in \mathcal{J} \\
& S_j + p_j \leq S_i && \forall (j,i) \in \mathcal{E} \\
& S_j \geq S_i + p_i - M \cdot z_{ji} && j,i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K} \\
& S_i \geq S_j + p_j - M \cdot (1 - z_{ji}) && j,i \in \mathcal{J}_k, j < i, \forall k \in \mathcal{K} \\
& 0 \leq S_j \leq |\mathcal{H}| && \forall j \in \mathcal{J} \\
& z_{ji} \in \{0,1\} && \forall j,i \in \mathcal{J}_k, \forall k \in \mathcal{K}
\end{aligned}
$$

C_{max} is the largest end-time

Precedence constraints

Resource constraints

Where:

- $\mathcal{J}_k$ is the set of operations on resource $k \in \mathcal{K}$
- $\mathcal{E}$ is the set of all operation pairs (j,i) s.t. i is constrained to be after j
- M a big constant
- S_j is the start-time of operation j
- $z_{ji} = 1$ iff operation i is scheduled after operation j

Time-Indexed vs. Disjunctive

Problem	Disjunctive		Time-Indexed	
	Time (geo/arith)	Opt	Time (geo/arith)	Opt
3 × 3	0.01/0.01	10	0.04/0.04	10
4 × 3	**0.01/0.01**	10	0.07/0.07	10
5 × 3	0.02/**0.02**	10	0.22/0.22	10
3 × 6	**0.01/0.01**	10	0.16/0.17	10
3 × 8	**0.01/0.01**	10	0.23/0.24	10
3 × 10	**0.01/0.01**	10	0.55/0.56	10
5 × 5	**0.02/0.02**	10	1063.03/2003.15	5
8 × 8	**0.60/0.61**	10	1986.61/2518.23	5
10 × 10	**3.19/3.48**	10	-[9]	-
12 × 12	**99.43/212.76**	10	-[10]	-
15 × 15	**1568.89/2272.75**	5	#	#
20 × 15	-	-	#	#

[Ku & Beck, 2016] Solver: CPLEX. Time-limit: 3600 s. Memory-limit: 8 GB
‘-‘ no optimal solutions
‘-x’ no optimal solutions, no feasible in x instances
‘#’ model > 8 GB

With 10 years of development, these numbers would be a bit better today, but not a lot

Does this mean time-indexed is not useful?

MILP Summary

- A strong approach to scheduling problems
- Particularly good in timetabling (assigning resource to pre-defined slots)
- For many scheduling (and optimization) problems a MILP model is the place to start
  - often takes a significant amount of work to beat it

Constraint Programming (CP)

Constraint Satisfaction Problem (CSP)

- Given
  - V: a set of variables $\{v_1, \ldots, v_n\}$
  - D: a set of domains $\{D_1, \ldots, D_n\}$
  - C: a set of constraints $\{c_1, \ldots, c_m\}$
- Each constraint restricts the values that a subset of variables can take on (called the scope of a constraint)
- A constraint is a mapping from the the Cartesian product of the domains of the variables in its scope to $\{T, F\}$

Probably how you were taught CP

A Solution to a CSP

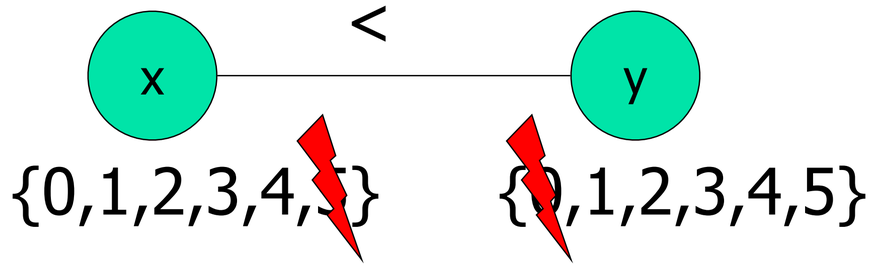
- A constraint is satisfied if the assignment of the variables in its scope maps to T
- In a solution:
  - each variable is assigned a value from its domain
    - $v_i = d_i, d_i \in D_i$
  - each constraint is satisfied
- A constraint optimization problem (COP) is a CSP plus an objective function $f(V) \rightarrow \mathbb{R}$ which is maximized or minimized

Solving a CSP (or COP)

- Tree search by assigning values to variables
- At each node in the search tree, perform propagation
  - propagation = inference
    - most commonly: remove values from domains of variables that can be proved to not take part in a solution (given the assignments already made)
- COP: when a feasible solution is found add a new constraint that is satisfied only if the objective is better than the incumbent

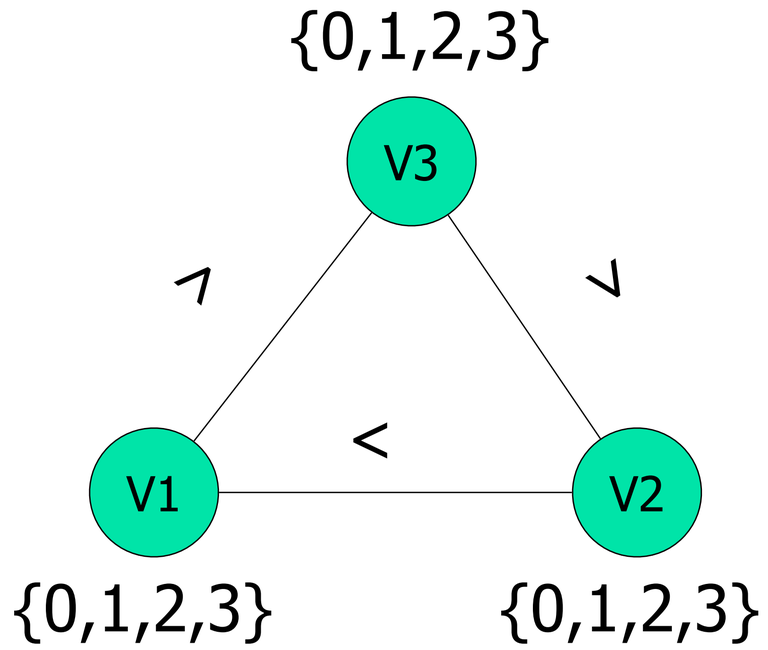
Arc Consistency (AC)

- Fundamental form of propagation
- A binary constraint, $c(x, y)$, is arc consistent iff
  - for all values $v_x \in D_x$ there exists a value $v_y \in D_y$ such that $c(x = v_x, y = v_y) \rightarrow T$
  - and similarly for all values $v_y \in D_y$



AC on a Problem

- A problem is AC if each of its constraints is AC
- Enforce AC on the following constraint network



The Locality of Inference

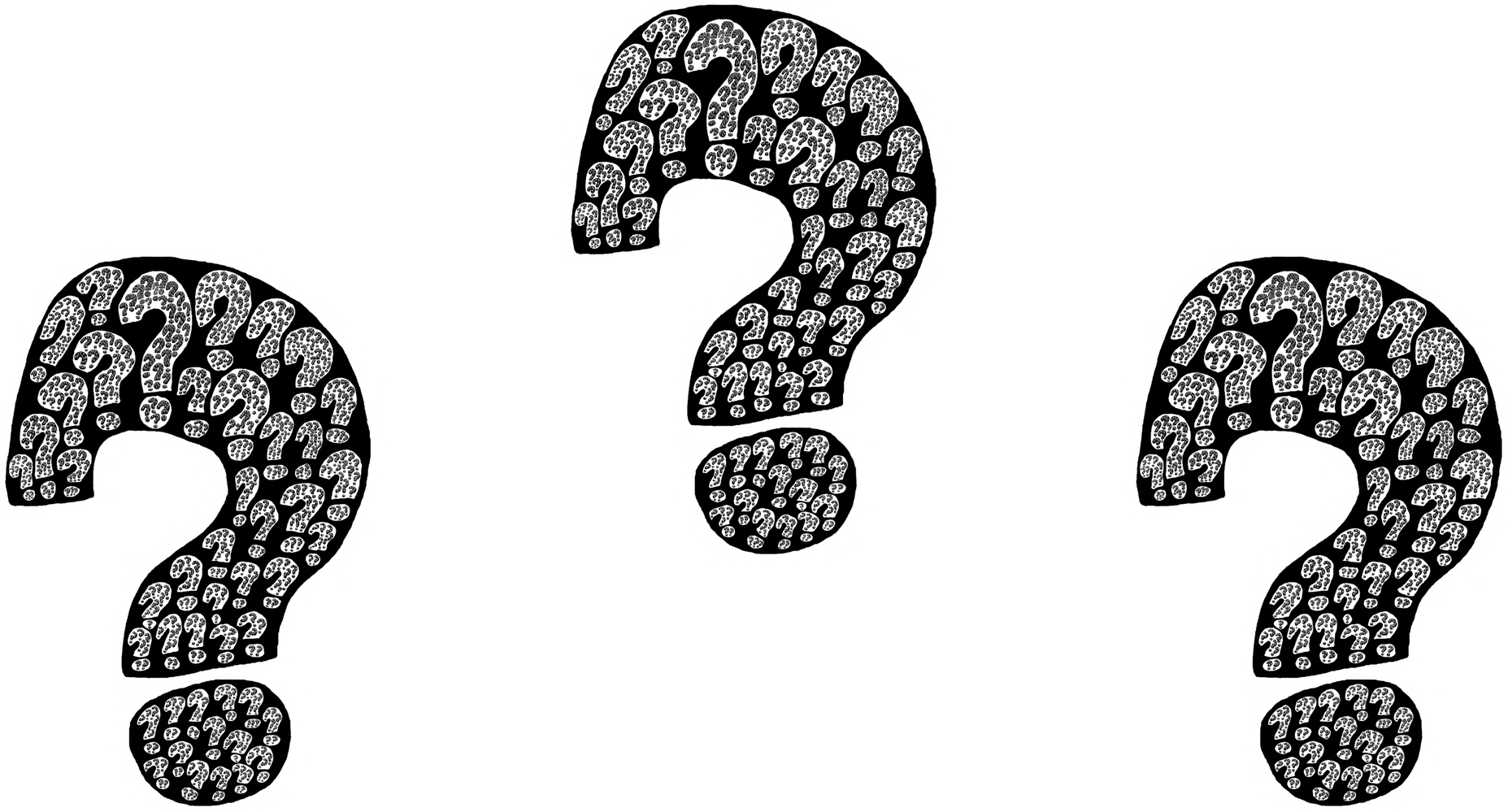
- Tree search by assigning values to variables
  - and we have heuristics to choose variables and values
- But the main computational engine of CP is
  inference inside a constraint
- Global consistency: all domain values are part of at least one global solution
  - (as hard as solving the whole problem)
- AC: all domain values are part of an assignment that satisfies each constraint on the variable: local consistency

The Locality of Inference

This is a key to understanding what comes next

- Efficiently make sound inferences by reasoning about one constraint at a time
- Constraints "communicate" via domain pruning
  - constraint $A(x, y)$ enforces AC and removes a value from, let's say, variable x
  - constraint $B(x, z)$ re-enforces AC wrt new domain of x
  - … inference propagates across the constraint network!

Questions?



By Khaydock - https://commons.wikimedia.org/w/index.php?curid=25102900

Modern Constraint Programming

- You've just seen the textbook definition of constraint programming
- It's correct but does not really reflect a modern perspective, which focuses on
  - rich variable types (not everything is an integer)
  - "global" constraints

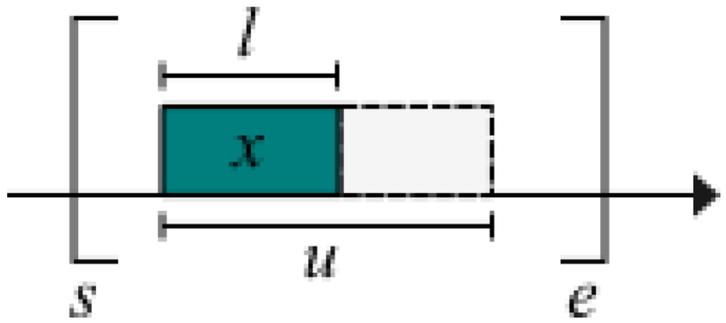
Extensible Variable Types

- MILP is restricted to integer and continuous variables
- SAT is restricted to boolean variables
- CP has no such restrictions
  - Researchers can extend the “language” by adding new variable types: sets, graphs, intervals, functions, …
- So what is a variable?
  - An object with a domain of values, one of which must be assigned to form a solution

Strange New Variable #1: Interval Variables

- A variable whose value is a half open interval

$$\text{Domain: } \{[s, e) \mid s, e \in \mathbb{Z}, s \leq e\}$$



- Useful for scheduling(!) but also other problems such as cutting-stock problems [Luo & Beck CPAIOR2022]

But it gets worse …

Optional Interval Variables

- We extend the domain with a distinct value, $\perp$, which indicates that the interval is not present

$$\{ \perp \} \cup \{[s, e) \,|\, s, e \in \mathbb{Z}, s \leq e\}$$

Variable is not present

- Any constraint on an “absent” interval variables must ignore it

OK … so?

In order to get the point of this, we need to talk about global constraints

Global Constraints

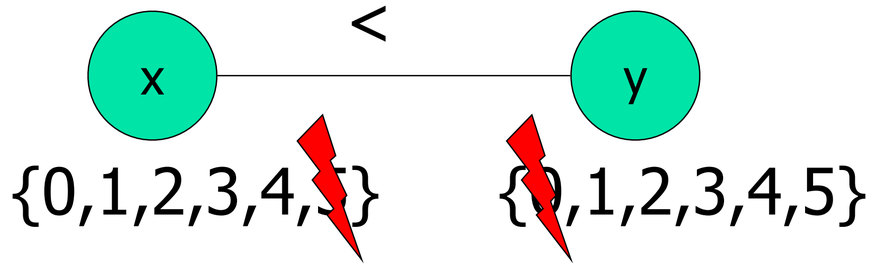
- A confusing concept because it does not have a crisp mathematical definition
- A global constraint is a restriction over the feasible assignments of a set of variables (with parametrizable cardinality)
  - Not just “binary” constraints but k-ary for a user-defined k
- Let $X = \{x_1, \ldots, x_k\}$ be a set of integer variables
- The global constraint all-diff(X) is satisfied iff all $x_i \in X$ are assigned pairwise different values

Combinatorial Substructure and Inference

- A global constraint provides a more global perspective on a combinatorial substructure
- We still need to reason locally about one constraint
  - but now the constraint represents meaningful structure in the problem

Arc Consistency (AC)

- A binary constraint, $c(x, y)$, is arc consistent iff
  - for all values $v_x \in D_x$ there exists a value $v_y \in D_y$ such that $c(x = v_x, y = v_y) \rightarrow T$
  - and similarly for all values $v_y \in D_y$



Generalized Arc Consistency (GAC)

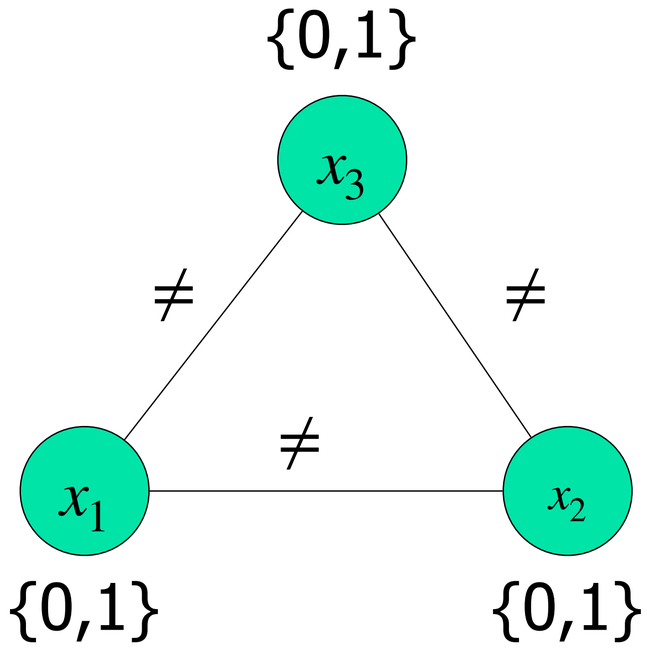
- A constraint, $c(x_1, \ldots, x_k)$, is generalized arc consistent iff for all variables $x_i, i \in \{1,\ldots,k\}$ and for all values $v_{x_i} \in D_{x_i}$ there exists a tuple of values $[v_{x_j} \in D_{x_j}], i \neq j$ such that

$$c(x_i = v_{x_i}, [x_j = v_{x_j}]) \rightarrow T$$

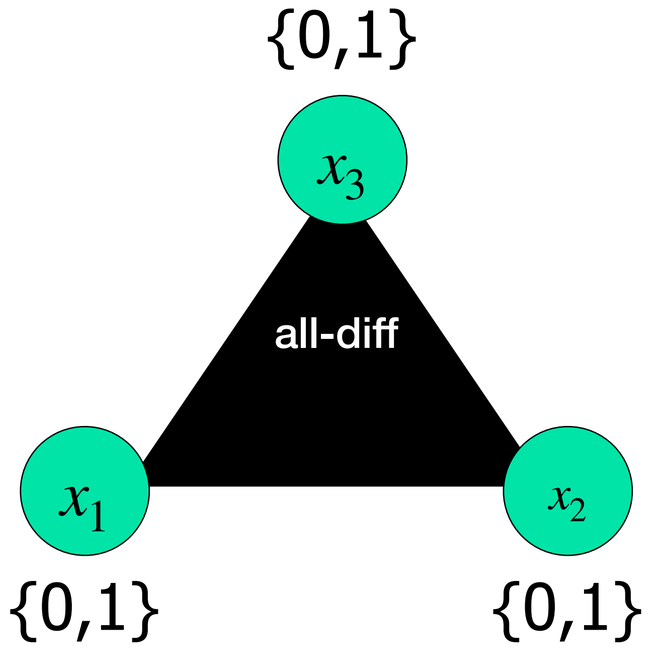
- Each value in each domain must be part of at least one solution to the constraint

Also called “domain consistency”

(G)AC on a Problem



Enforce AC



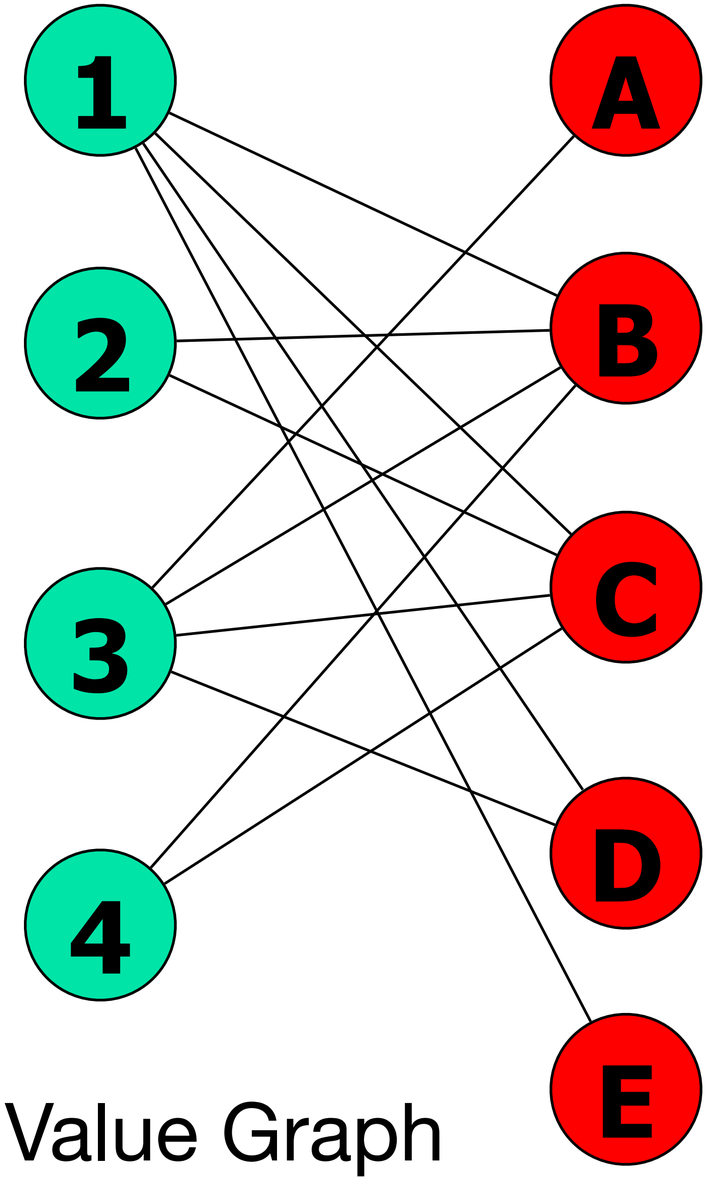
Enforce GAC

We can prove that GAC on all-diff is stronger than AC on a clique of ≠

Complexity of GAC on All-Diff?

- $x_1 \in \{B, C, D, E\}$
- $x_2 \in \{B, C\}$
- $x_3 \in \{A, B, C, D\}$
- $x_4 \in \{B, C\}$
- all-diff$(x_1, \ldots, x_4)$

Thm (Regin AAAI94): all-diff$(x_1, \ldots, x_k)$ is GAC iff every edge in the value graph belongs to a matching covering $\{x_1, \ldots, x_k\}$



Value Graph

Global Constraint

- A constraint over an arbitrary number of variables
- Representing a recurring combinatorial substructure
  - “recurring” so that it is useful to model many problems
- With accompanying inference algorithm(s) to efficiently enforce GAC
  - (sometimes GAC is too expensive so we settle for something weaker)

Optional Interval Variables

- We extend the domain with a distinct value, $\perp$, which indicates that the interval is not present

$$\{ \perp \} \cup \{[s, e) \,|\, s, e \in \mathbb{Z}, s \leq e\}$$

Variable is not present

- Any constraint on an “absent” interval variables must ignore it

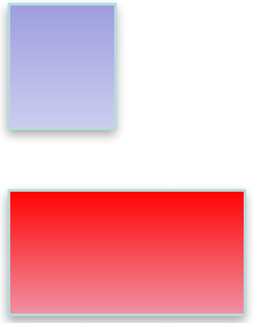
OK … so?

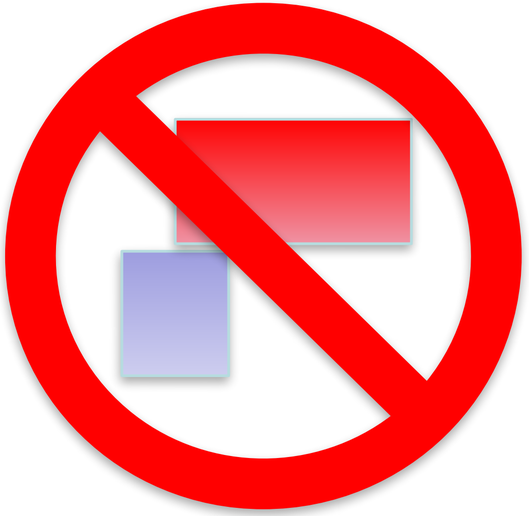
In order to get the point of this, we need to talk about global constraints

Global Constraints on Interval Variables

- X is a set of (optional) interval variables
- no-overlap(X)
- intervals must have empty intersections

GAC is NP-complete

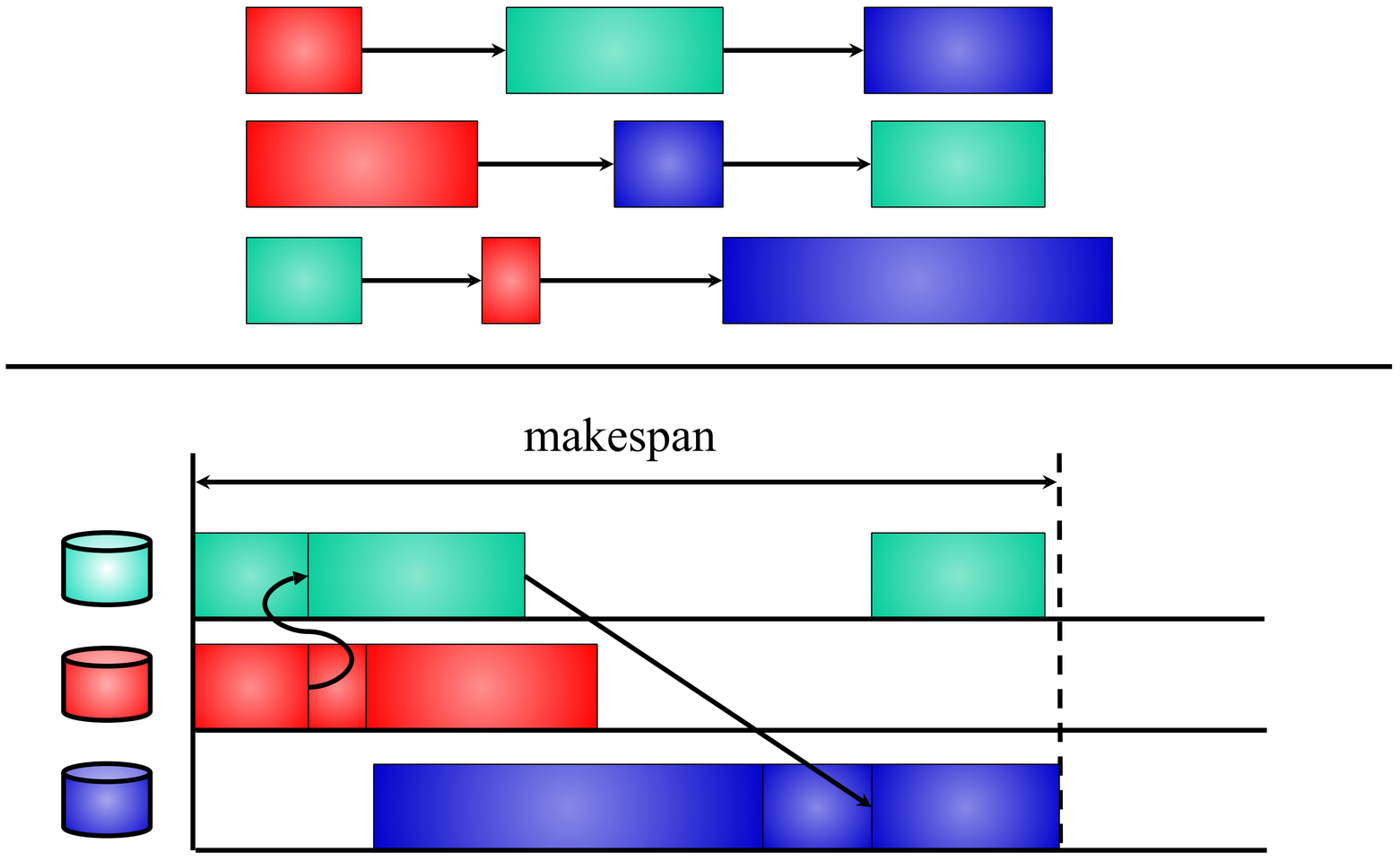




- $y \notin X$ is an (optional) interval variable
- alternative(y, X)
- y is present iff one $x \in X$ is present and start(x) = start(y) and end(x) = end(y)

Both of these constraints are useful for scheduling

The Job Shop Scheduling Problem (JSP)



CP Model for JSP

$$\min \quad \max_{j \in J}(\textsc{End}(x_{\sigma^j_m, j}))$$

Minimize makespan

$$\textsc{Pres}(x_{i,j}) = 1 \qquad \forall i \in M, j \in J$$

All interval vars present

$$\textsc{Length}(x_{i,j}) = p_{i,j} \qquad \forall i \in M, j \in J$$

Processing times

$$\textsc{EndsBeforeStart}(x_{\sigma^j_{h-1}, j}, x_{\sigma^j_h, j}) \qquad \forall j \in J, 2 \leq h \leq m$$

Precedence constraints

$$\textsc{NoOverlap}(\{x_{i,j} : j \in J\}) \qquad \forall i \in M$$

Resource constraints

$$x_{i,j} : \textsc{IntervalVar}() \qquad \forall i \in M, j \in J$$

- J is the set of jobs, $n = |J|$
- M is the set of machines, $m = |M|$
- $x_{i,j}$ is an interval variable in job j that requires machine i
- σ^j_h is the machine required by the hth operation of job j

MILP vs. CP

Problem	Disjunctive		Time-Indexed		CP	
	Time (geo/arith)	Opt	Time (geo/arith)	Opt	Time (geo/arith)	Opt
3 × 3	0.01/0.01	10	0.04/0.04	10	**0.00/0.00**	10
4 × 3	0.01/0.01	10	0.07/0.07	10	**0.00/0.00**	10
5 × 3	0.02/0.02	10	0.22/0.22	10	**0.00/0.00**	10
3 × 6	0.01/0.01	10	0.16/0.17	10	**0.00/0.00**	10
3 × 8	0.01/0.01	10	0.23/0.24	10	**0.00/0.00**	10
3 × 10	0.01/0.01	10	0.55/0.56	10	**0.00/0.00**	10
5 × 5	0.02/0.02	10	1063.03/2003.15	5	**0.00/0.00**	10
8 × 8	0.60/0.61	10	1986.61/2518.23	5	**0.15/0.15**	10
10 × 10	3.19/3.48	10	-[9]	-	**0.67/0.68**	10
12 × 12	99.43/212.76	10	-[10]	-	**3.58/3.88**	10
15 × 15	1568.89/2272.75	5	#	#	**39.57/47.71**	10
20 × 15	-	-	#	#	**1037.81/1924.35**	6

[Ku & Beck, 2016] Solvers: CPLEX, CPO. Time-limit: 3600 s. Memory-limit: 8 GB

‘-‘ no optimal solutions

‘-x’ no optimal solutions, no feasible in x instances

‘#’ model > 8 GB

CP Summary

- A very strong approach to scheduling problems
  - Often beats MILP but less popular
  - Interest in CP scheduling outside CP/AI community is growing
- Tree search + inference
  - arc consistency and generalized arc consistency
- Rich variables and global constraints

Constraints vs. States

- MILP and CP are constraint-based formalisms
- Planning and heuristic search → state-based
- Very little work in state-based approaches to scheduling
  - or optimization in general
- Why?

Can we use A*-style search for scheduling?

A State-based Model for $1||\sum w_j T_j$?

NP-hard (knapsack)

- 1-machine
- $T_j = \max(0, C_j - d_j)$: due date: d_j
- F: the set of scheduled operations
- P_i: the set of predecessors of i (preprocessing)

compute $V(\emptyset)$

$$V(F) = \begin{cases} 0 & \text{if } F = N \\ \min_{i \in N \setminus F : P_i \subseteq F} w_i T(i, F) + V(F \cup \{i\}) & \text{else} \end{cases}$$

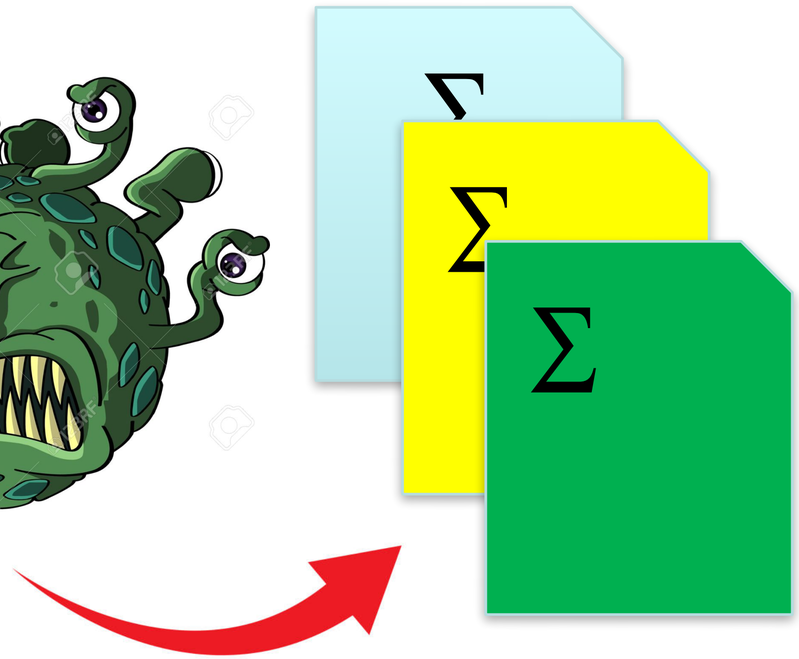
$V(F) \geq 0.$

$$T(i, F) = \max(0, \sum_{j \in F} p_j + p_i - d_i)$$

Model-and-Solve

Problem

Problem Definition



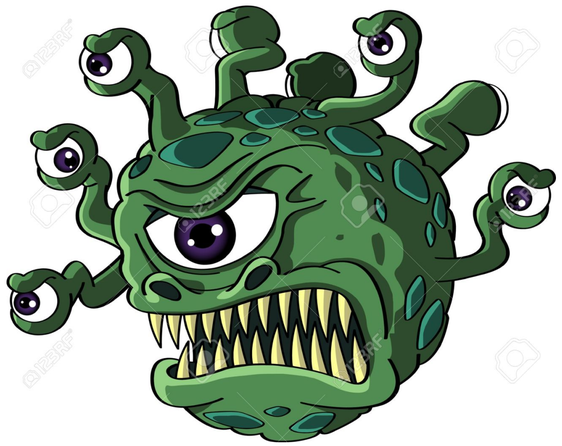
Model-and-Solve

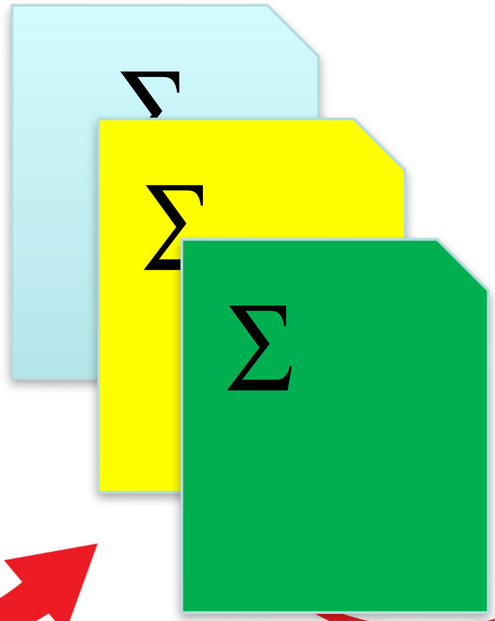
Problem

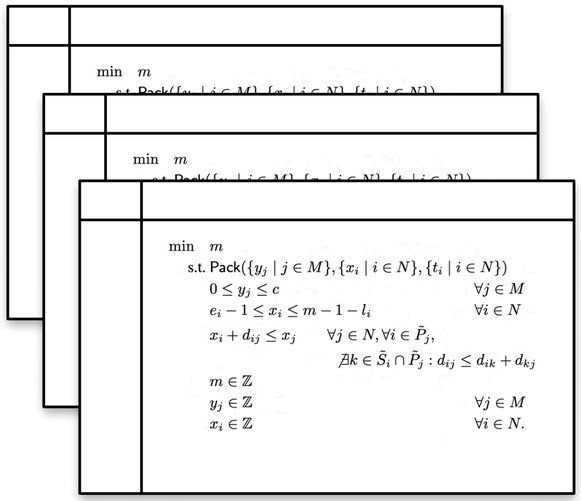
Problem Definition

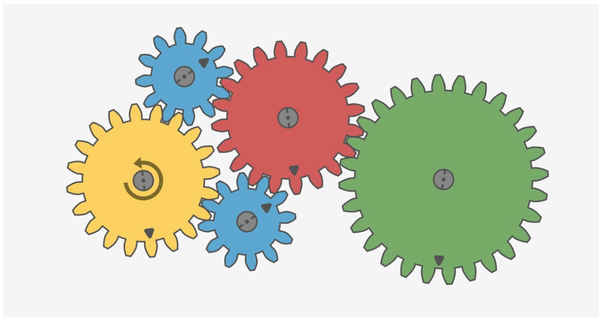
Models

General Purpose Solver









CP, LP, MIP, MINLP, AI Planning, …

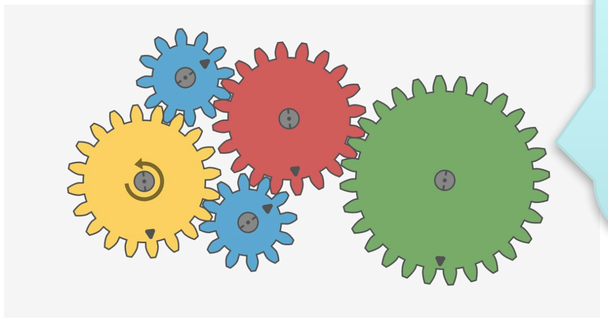
A Solution!

Model-and-Solve for Dynamic Programming

- Domain-Independent Dynamic Programming (DIDP)

Define models using DP transition system

$$
\begin{aligned}
&\min \quad m \\
&\text{s.t. } \mathsf{Pack}(\{y_j \mid j \in M\}, \{x_i \mid i \in N\}, \{t_i \mid i \in N\}) \\
&\quad 0 \leq y_j \leq c && \forall j \in M \\
&\quad e_i - 1 \leq x_i \leq m - 1 - l_i && \forall i \in N \\
&\quad x_i + d_{ij} \leq x_j \quad \forall j \in N, \forall i \in \tilde{P}_j, \\
&\qquad\qquad \nexists k \in \tilde{S}_i \cap \tilde{P}_j : d_{ij} \leq d_{ik} + d_{kj} \\
&\quad m \in \mathbb{Z} \\
&\quad y_j \in \mathbb{Z} && \forall j \in M \\
&\quad x_i \in \mathbb{Z} && \forall i \in N.
\end{aligned}
$$



Solve models using heuristic state-based search

https://didp.ai

But What is the Heuristic ($h(s)$)?

- Models define a dual bound on states, $\eta(s)$
- Used for pruning vs. incumbent in any-time state-based search
- Can be used for heuristics guidance i.e. $h(s) = \eta(s)$
- Can also use learned heuristics [Narita et al. CPAIOR2025] and inferred heuristics (e.g. generalization of operator counting [Singh et al. ICAPS2026]

compute $V(\emptyset)$

$$V(F) = \begin{cases} 0 & \text{if } F = N \\ \min_{i \in N \setminus F : P_i \subseteq F} w_i T(i, F) + V(F \cup \{i\}) & \text{else} \end{cases}$$

$V(F) \geq 0.$

$$T(i, F) = \max(0, \sum_{j \in F} p_j + p_i - d_i)$$

DIDP vs. MILP and CP

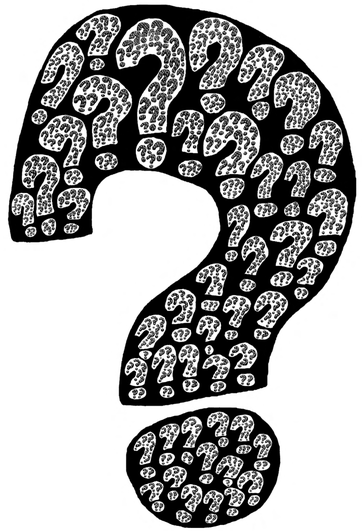
[Kuroiwa & Beck, *AIJ*, 2026]

	MIP		CP		CAASDy		DFBnB		CBFS		ACPS		APPS		DBDFS		CABS	
	c.	m.	c.	m.	c.	m.	c.	m.	c.	m.	c.	m.	c.	m.	c.	m.	c.	m.
TSPTW (340)	224	0	47	0	**257**	83	**242**	34	**257**	81	**257**	82	**257**	83	**256**	83	**259**	0
CVRP (207)	**28**	5	0	0	6	201	6	187	6	201	6	201	6	201	6	201	6	0
m-PDTSP (1178)	940	0	**1049**	0	952	226	985	193	988	190	988	190	988	190	987	191	1035	0
OPTW (144)	16	0	49	0	**64**	79	**64**	60	**64**	80	**64**	80	**64**	80	**64**	78	**64**	0
MDKP (276)	**168**	0	6	0	4	272	4	272	5	271	5	271	5	271	4	272	5	1
Bin Packing (1615)	1160	0	**1234**	0	922	632	526	1038	1115	431	1142	405	1037	520	426	1118	1167	4
SALBP-1 (2100)	1431	250	1584	0	**1657**	406	**1629**	470	1484	616	**1626**	474	**1635**	465	1404	696	**1802**	0
$1\|\|\sum w_i T_i$ (375)	107	0	150	0	**270**	105	**233**	8	**272**	103	**272**	103	**265**	110	**268**	107	**288**	0
Talent Scheduling (1000)	0	0	0	0	**207**	793	**189**	388	**214**	786	**214**	786	**206**	794	**205**	795	**239**	0
MOSP (570)	241	14	437	0	**483**	87	**524**	46	**523**	47	**524**	46	**523**	47	**522**	48	**527**	0
Graph-Clear (135)	26	0	4	0	**78**	57	**99**	36	**101**	34	**101**	34	**99**	36	**82**	53	**103**	19

Coverage, 30 minutes, 8 GB

Not (yet) mainstream for scheduling but we will see it again in the second half

The ~~Plan~~ Schedule



- Part 1: The Basics
  - What is scheduling, where is studied, computational complexity, primary solution techniques
- Part 2: Solving Three Scheduling Problems
  - Single-machine with inventory, Assembly-line balancing, Multi-project scheduling

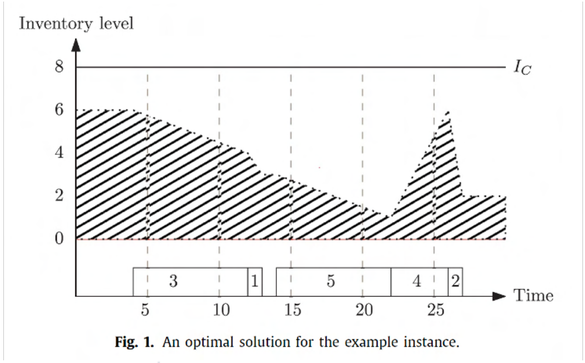
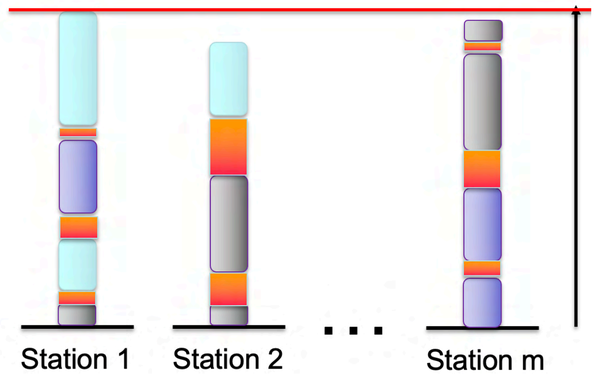
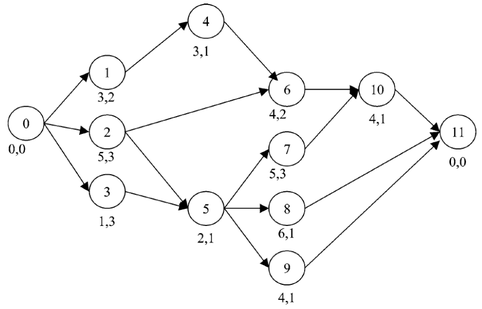
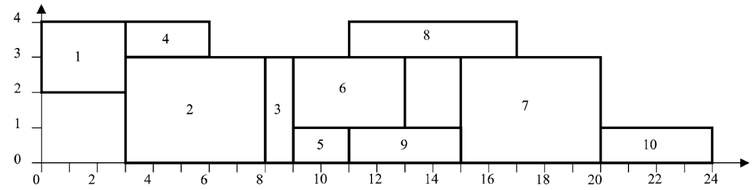


Fig. 1. An optimal solution for the example instance.







Part 2: Solving Three Scheduling Problems

The ~~Plan~~ Schedule

- Part 1: The Basics
  - What is scheduling, where is studied, computational complexity, primary solution techniques
- Part 2: Solving Three Scheduling Problems
  - Single-machine with inventory, Assembly-line balancing, Multi-project scheduling

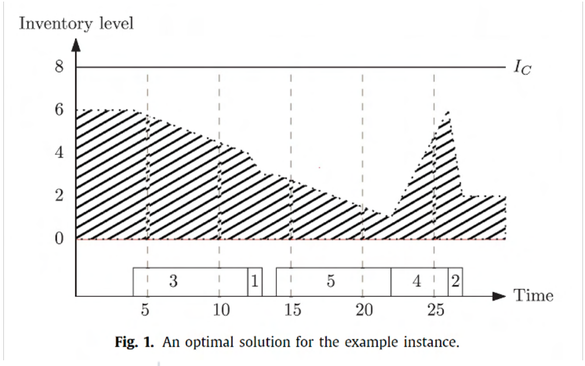
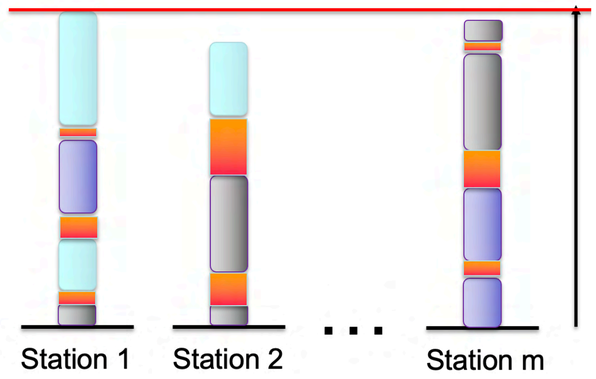
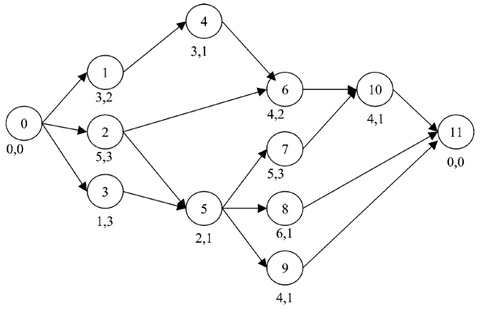
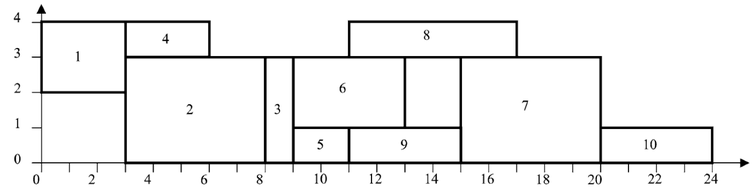


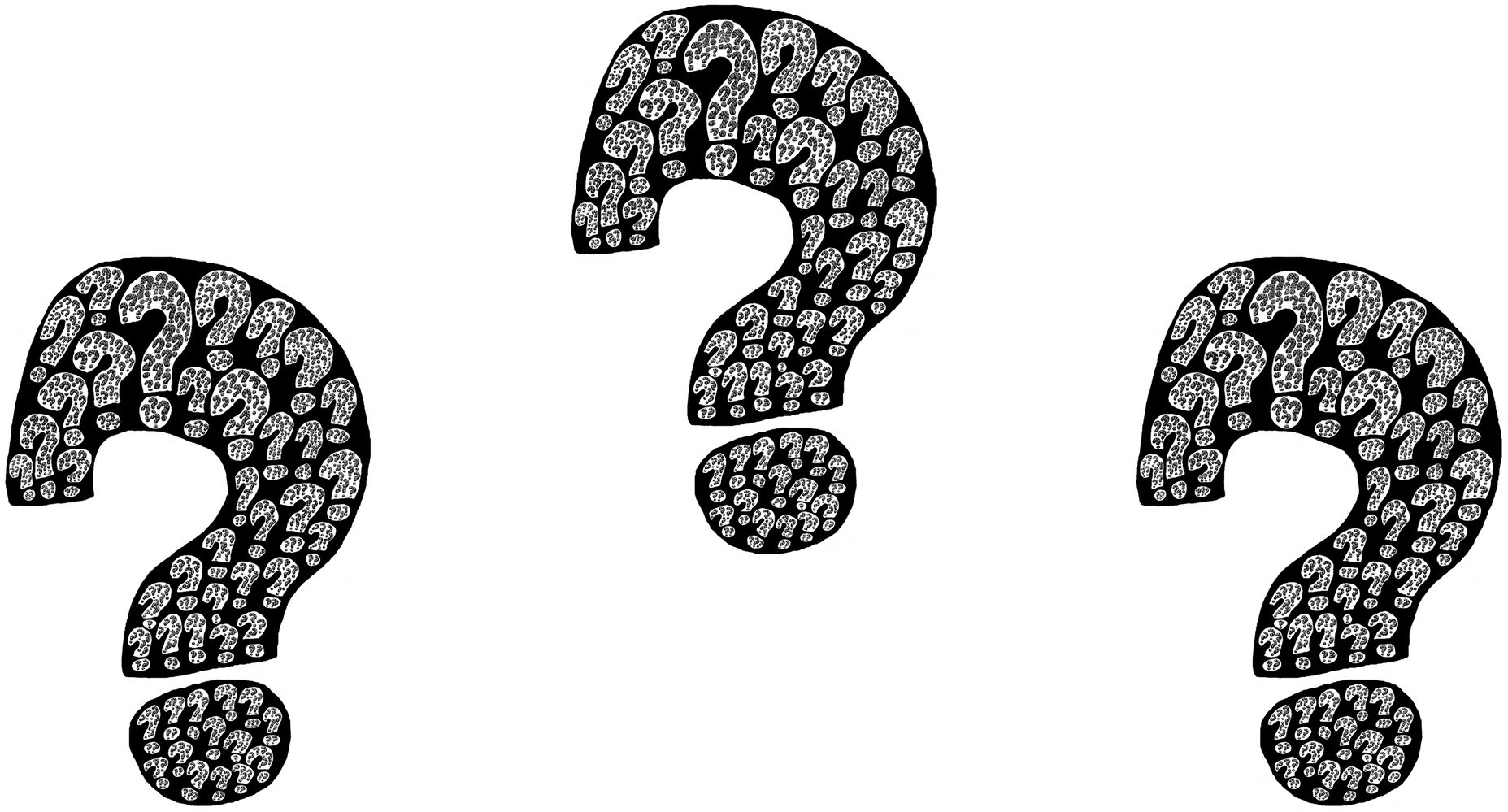
Fig. 1. An optimal solution for the example instance.







Questions?



By Khaydock - https://commons.wikimedia.org/w/index.php?curid=25102900

P1: Inventory Scheduling ($1 \mid inv, r_j \mid C_{max}$)

[Davari et al., *EJOR*, 2020]

- Single machine with release dates and inventory
- Minimize makespan
- Each operation j has a
  - processing time, p_j
  - release date, r_j
  - inventory contribution, δ_j (postive or negative)

Table 1
Data for the example instance.

j	1	2	3	4	5
p_j	1	1	8	4	8
r_j	7	1	4	18	14
δ_j	−1	−4	−2	5	−2

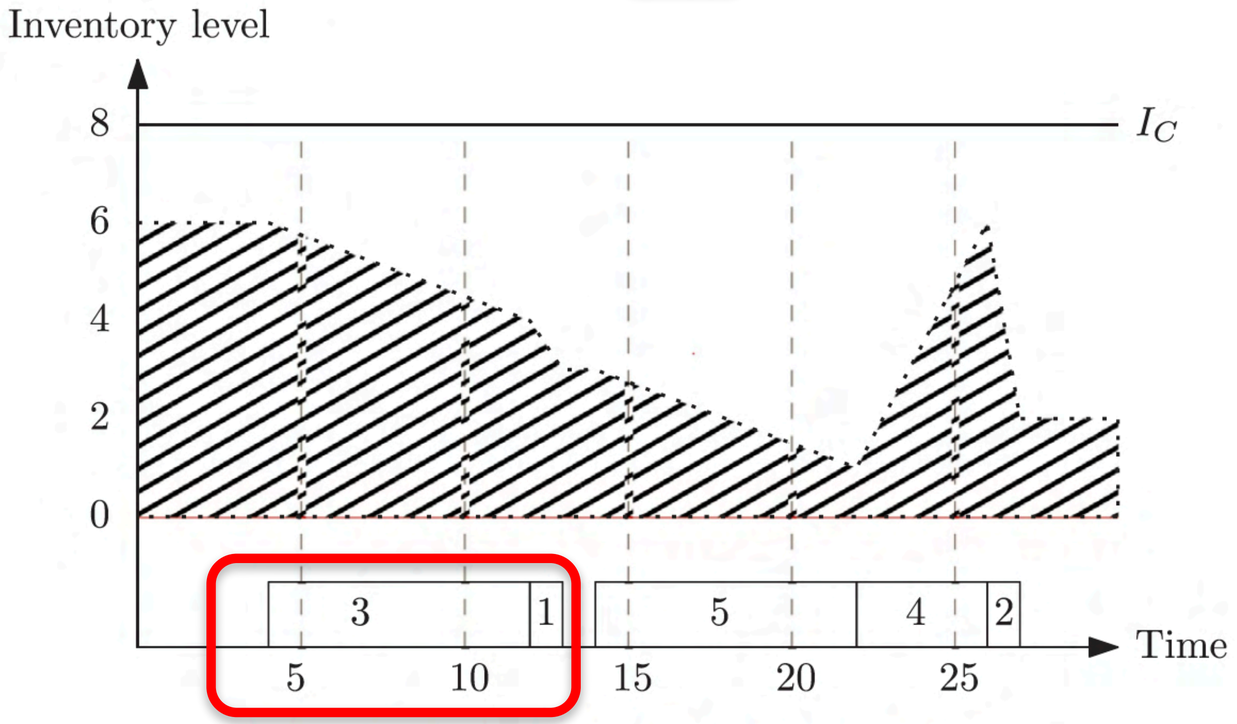


Fig. 1. An optimal solution for the example instance.

P1: MILP Formulation

x_{js} = 1 if operation j is in slot s

z_s: completion time of operation in slot s

SBF : min z_n

subject to

$$\sum_{s=1}^{n} x_{js} = 1 \qquad j \in J \qquad (7)$$

$$\sum_{j \in J} x_{js} = 1 \qquad s = 1, \ldots, n \qquad (8)$$

one operation per slot, one slot per operation

$$z_s \geq \sum_{j \in J} x_{js}(r_j + p_j) \qquad s = 1, \ldots, n \qquad (9)$$

$$z_s \geq z_{s-1} + \sum_{j \in J} x_{js} p_j \qquad s = 2, \ldots, n \qquad (10)$$

compute completion times

$$y_1 = I_0 + \sum_{j=1}^{n} \delta_j x_{j1} \qquad (11)$$

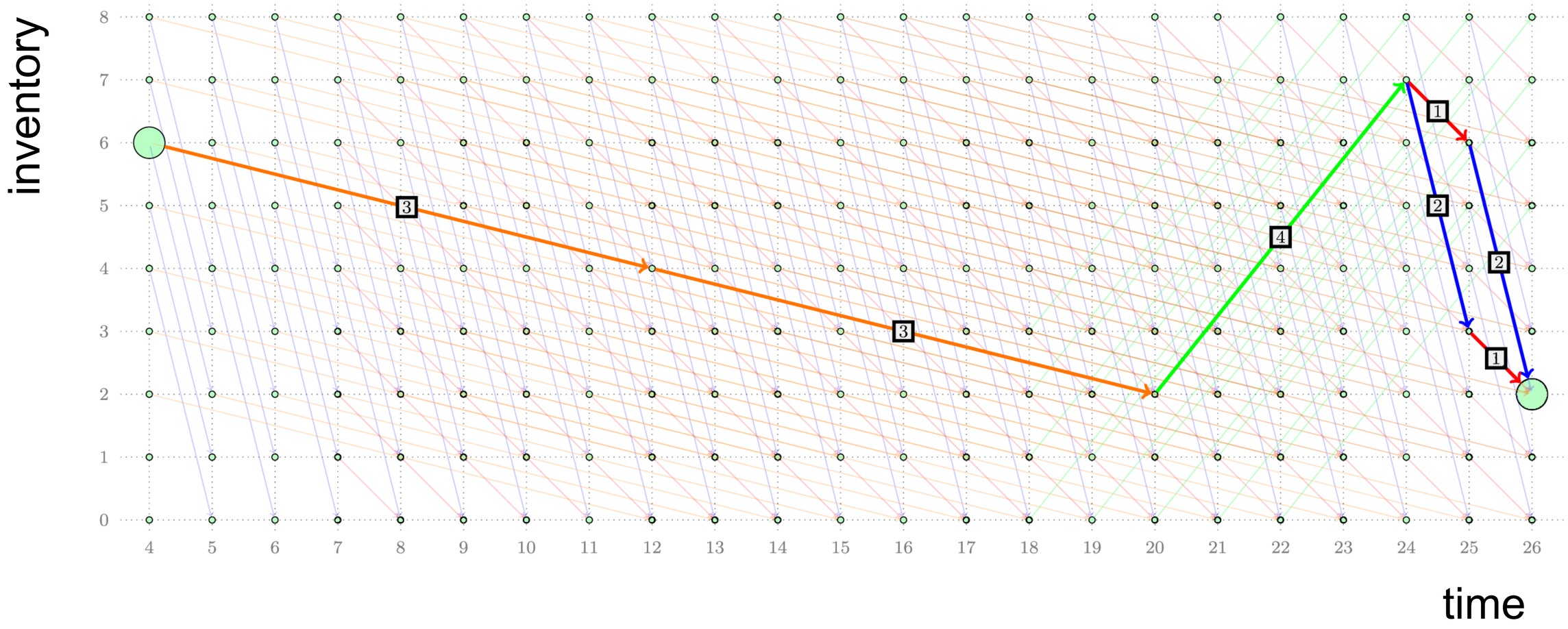
$$y_s = y_{s-1} + \sum_{j=1}^{n} \delta_j x_{js} \qquad s = 2, \ldots, n \qquad (12)$$

$$0 \leq y_s \leq I_C \qquad s = 1, \ldots, n \qquad (13)$$

compute & constrain inventory level

[Davari et al., *EJOR*, 2020]

P1: Complex DP Formulation



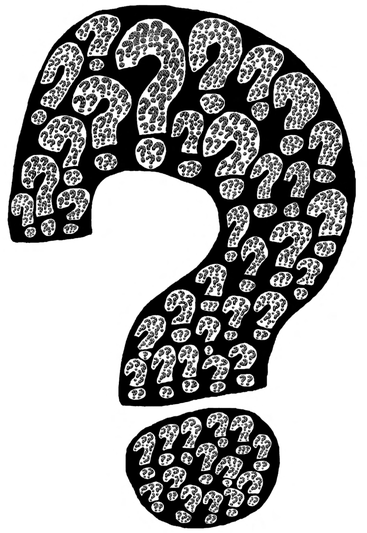
P1: Results

- # instances optimally solved in 1000s
- BB: custom branch-and-bound
- 96 instances per row
  - '-' did not run

Where are we going?
- **CP model**
- **DIDP model**

[Davari et al., *EJOR*, 2020]

n	MILP	BB	DP
10	96	96	96
20	96	96	96
30	92	79	96
40	75	70	93
50	54	62	89
60	-	-	80
70	-	-	72
80	-	-	74
90	-	-	65
100	-	-	60
Total	-	-	821



CP Summary

- A very strong approach to scheduling problems
  - Often beats MILP but less popular
  - Interest in CP scheduling outside CP/AI community is growing
- Tree search + inference
  - arc consistency and generalized arc consistency
- Rich variables and global constraints

Optional Interval Variables

- We extend the domain with a distinct value, $\perp$, which indicates that the interval is not present

$$\{ \perp \} \cup \{ [s, e) \mid s, e \in \mathbb{Z}, s \leq e \}$$

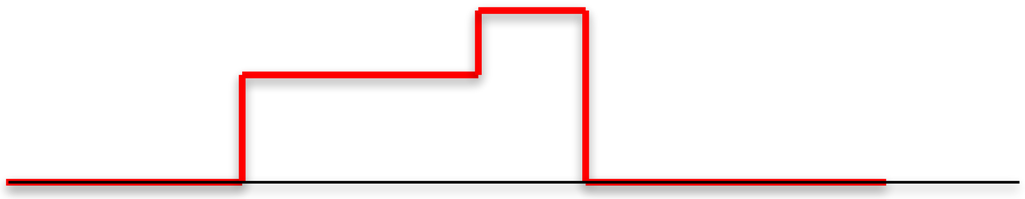
Variable is not present

- Any constraint on an “absent” interval variables must ignore it

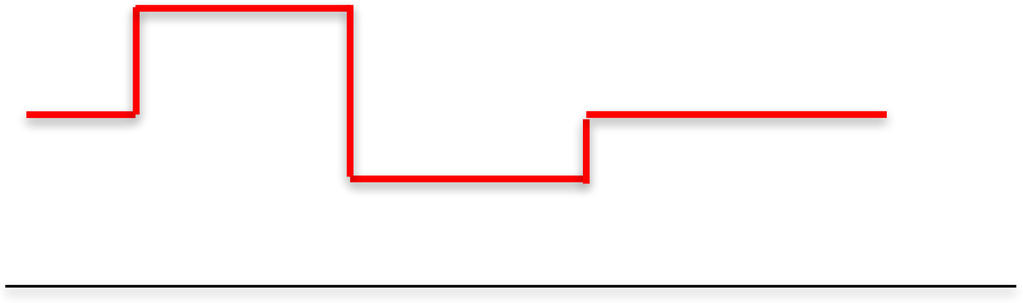
Strange New Variable #2: Cumulative Functions

- A variable whose value is a function over an integer domain

x =



y =



Building Cumulative Functions

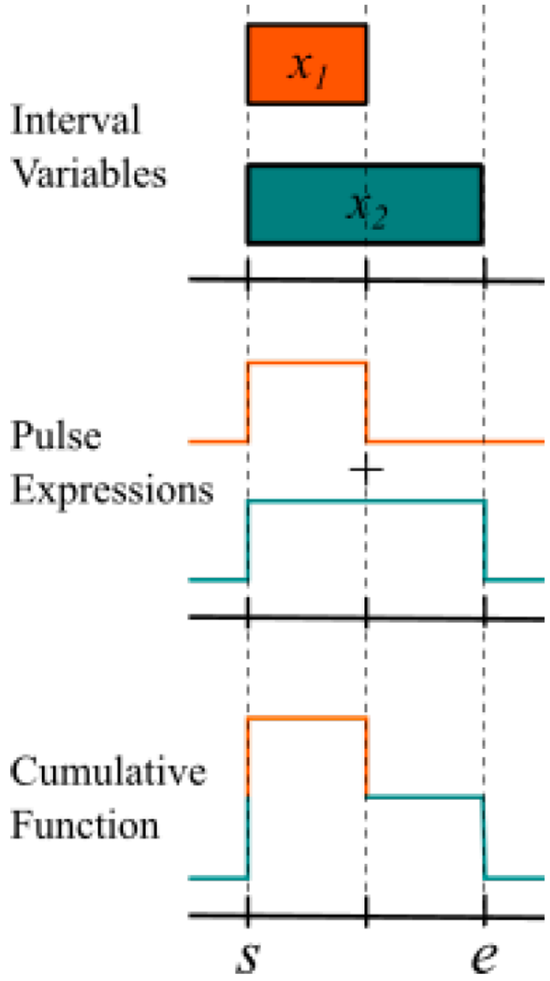
- A variable whose value is a function over an integer domain

x_1, x_2: interval vars

$f = pulse(x_1, 1)$

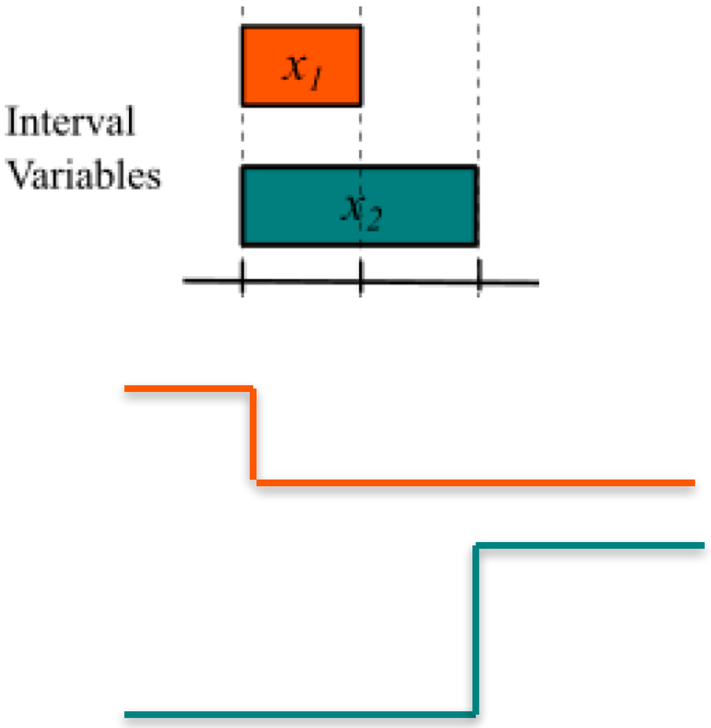
$g = pulse(x_2, 1)$

$h = f + g$



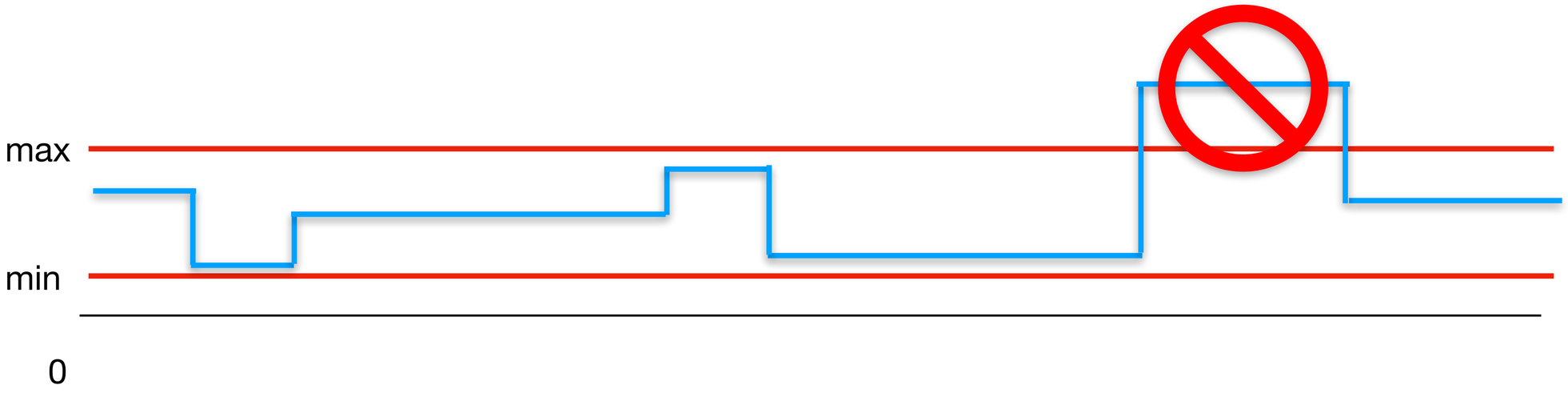
$f = stepAtStart(x_1, 1)$

$g = stepAtEnd(x_2, 2)$



Constraining Cumulative Functions

- AlwaysIn(f, min, max): the value of f must always be in $[min, max]$



P1: CP Formulation

$$\min \max_{j \in J} \mathsf{EndOf}(x_j)$$

minimize makespan

$$\mathsf{NoOverlap}(x_j | \forall j \in J)$$

sequence intervals

$$f = \mathsf{StepAtStart}(\mathsf{dummy}_s, I_0) + \sum_{j \in J} \mathsf{StepAtStart}(x_j, \delta_j)$$

$$\mathsf{AlwaysIn}(f, 0, I_C)$$

define/constrain cumulative function

$$x_j : \mathsf{intervalVar}(p_j, [r_j, \max_{i \in J} r_i + \sum_{i \in J} p_i])$$

$$\mathsf{dummy}_s : \mathsf{intervalVar}(1, [0, 1])$$

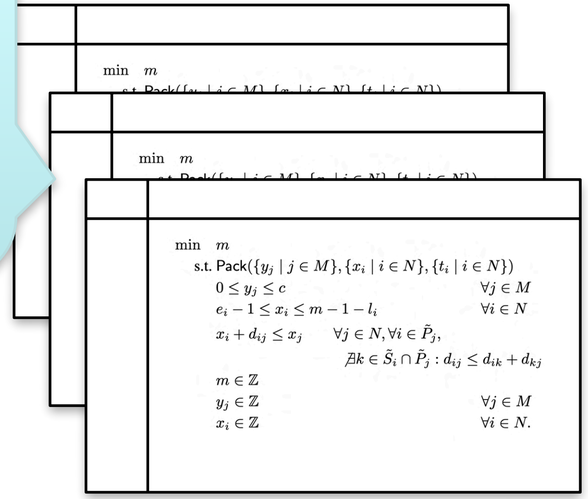
interval variables

$\forall j \in J$

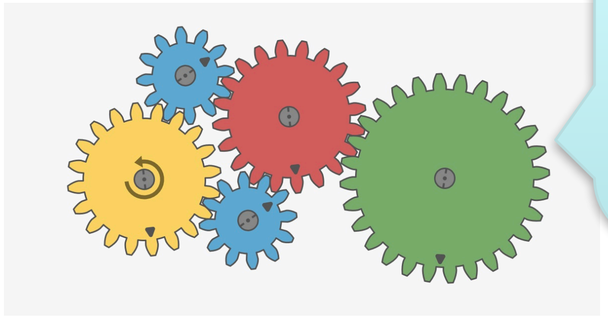
Model-and-Solve for Dynamic Programming

- Domain-Independent Dynamic Programming (DIDP)

Define models using DP transition system



$$
\begin{aligned}
\min \quad & m \\
\text{s.t. } & \mathsf{Pack}(\{y_j \mid j \in M\}, \{x_i \mid i \in N\}, \{t_i \mid i \in N\}) \\
& 0 \leq y_j \leq c && \forall j \in M \\
& e_i - 1 \leq x_i \leq m - 1 - l_i && \forall i \in N \\
& x_i + d_{ij} \leq x_j \quad \forall j \in N, \forall i \in \tilde{P}_j, \\
& \qquad \nexists k \in \tilde{S}_i \cap \tilde{P}_j : d_{ij} \leq d_{ik} + d_{kj} \\
& m \in \mathbb{Z} \\
& y_j \in \mathbb{Z} && \forall j \in M \\
& x_i \in \mathbb{Z} && \forall i \in N.
\end{aligned}
$$



Solve models using heuristic state-based search

https://didp.ai

Modeling for DIDP

- Need to define a state transition system
- State: a complete assignment to the state variables
- Transition: a modification to the state variables to produce a different state
- Solution: a sequence of transitions
- Base case: condition to stop the transitions

P1: DIDP Formulation

State variables:
- U : set of unexecuted jobs
- i : current inventory level
- t : current time

compute $V(J, I_0, 0)$

$$V(U,i,t) = \begin{cases} 0 & \text{if } U = \emptyset \\ \min_{j \in U} \max(0, r_j - t) + p_j + & \\ \qquad V(U \setminus \{j\}, i + \delta_j, \max(t, r_j) + p_j) & \text{if } U \neq \emptyset \wedge 0 \leq i + \delta_j \leq I^C \\ \infty & \text{otherwise} \end{cases}$$

$$V(U,i,t) \leq V(U,i,t') \qquad \text{if } t \leq t'$$

$$V(U,i,t) \geq \sum_{j \in U} p_i$$

$$V(U,i,t) \geq \max_{j \in J'}(r_j + \sum_{i=j}^{i=|J'|} p_i) - t$$

- J : set of all operations
- J' : J sorted in ascending r_j
- r_j : release time of j
- p_j: processing time of j
- δ_j : inventory contribution of j
- I_0: starting inventory
- I^C: inventory capacity

P1: Results

- # instances optimally solved in 1000s
- BB: custom branch-and-bound
- 96 instances per row
  - ‘-’ did not run

[Davari et al., *EJOR*, 2020]

n	MILP	BB	DP
10	96	96	96
20	96	96	96
30	92	79	96
40	75	70	93
50	54	62	89
60	-	-	80
70	-	-	72
80	-	-	74
90	-	-	65
100	-	-	60
Total	-	-	821

P1: Results

- # instances optimally solved in 1000s
- BB: custom branch-and-bound
- 96 instances per row
  - ‘-’ did not run

[Davari et al., *EJOR*, 2020]

n	MILP	BB	DP	DIDP
10	96	96	96	96
20	96	96	96	96
30	92	79	96	96
40	75	70	93	89
50	54	62	89	86
60	-	-	80	78
70	-	-	72	78
80	-	-	74	83
90	-	-	65	76
100	-	-	60	79
Total	-	-	821	857

P1: Results

- # instances optimally solved in 1000s
- BB: custom branch-and-bound
- 96 instances per row
  - ‘-’ did not run

[Davari et al., *EJOR*, 2020]

n	MILP	BB	DP	DIDP	CP
10	96	96	96	96	96
20	96	96	96	96	96
30	92	79	96	96	96
40	75	70	93	89	96
50	54	62	89	86	96
60	-	-	80	78	96
70			72	78	96
80			74	83	96
90			65	76	96
100			60	79	96
Total	-	-	821	857	960

mean CP run-time: 0.532s
max CP run-time: 20.473s

P1: Summary

- Simple DIDP model with heuristic search out-performs more complex DP algorithm
- Very simple CP model destroys everything else. Why?
  - Very strong inference from NoOverlap & AlwaysIn

edge-finding

$$\min \max_{j \in J} \mathsf{EndOf}(x_j)$$

$$\mathsf{NoOverlap}(x_j | \forall j \in J)$$

$$f = \mathsf{StepAtStart}(\mathsf{dummy}_s, I_0) + \sum_{j \in J} \mathsf{StepAtStart}(x_j, \delta_j)$$

$$\mathsf{AlwaysIn}(f, 0, I_C)$$

$$x_j : \mathsf{intervalVar}(p_j, [r_j, \max_{i \in J} r_i + \sum_{i \in J} p_i]) \qquad \forall j \in J$$

$$\mathsf{dummy}_s : \mathsf{intervalVar}(1, [0, 1])$$

P1: Summary

MILP

- 3 variable types
  - time-indexed
  - disjunctive
  - slot-based

CP

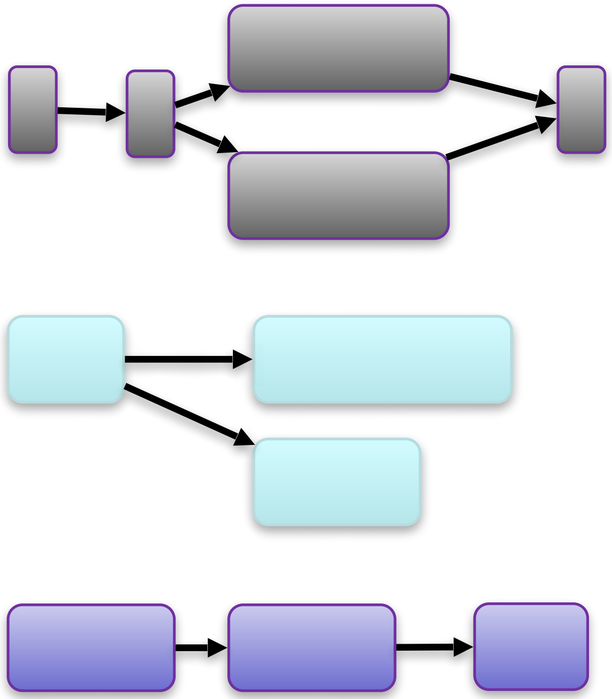
- interval variables
- cumulative functions
- strong propagation via edge-finding algorithms inside no-overlap and always-in

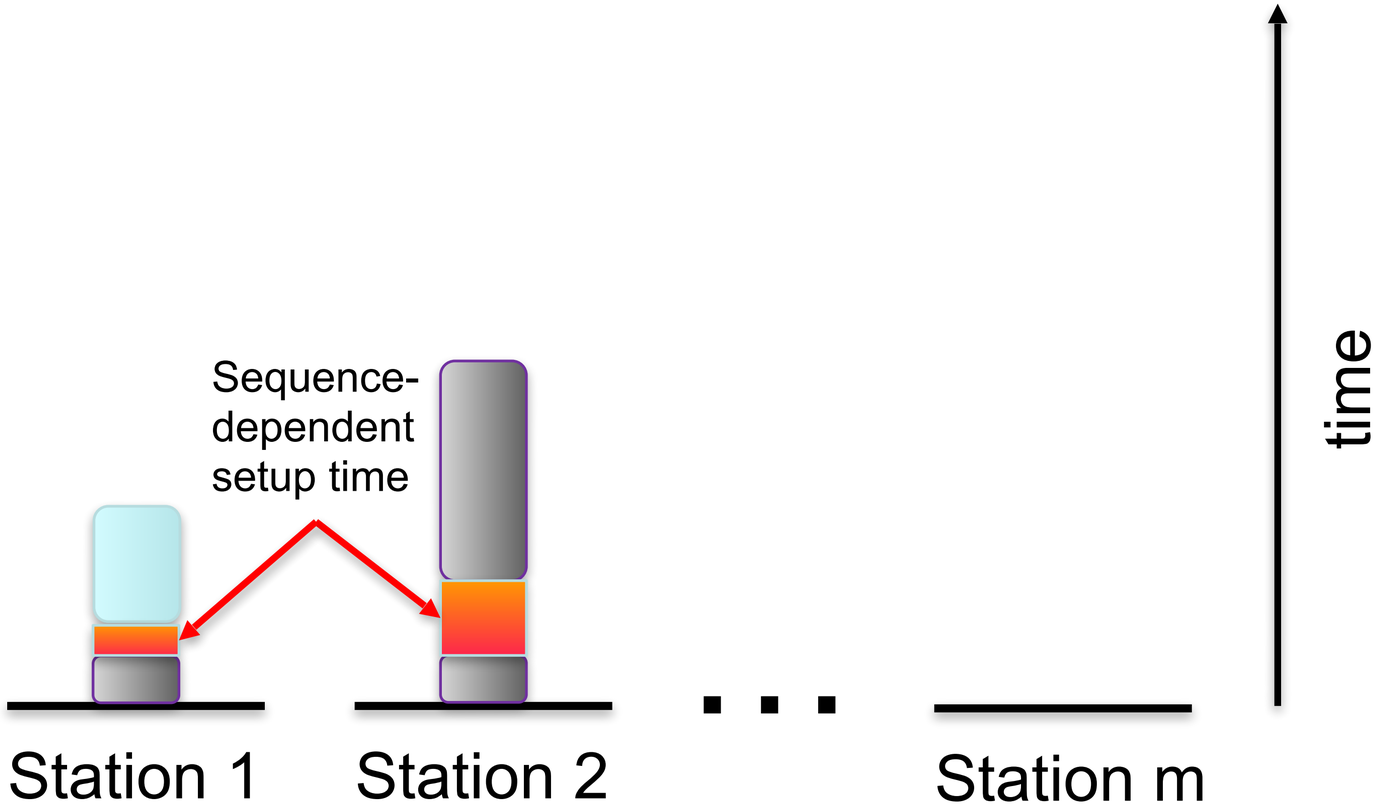
DIDP

- sequence of transitions = sequence of operations
- dual bounds

P2: Setup Assembly Line Balancing

[Zhang & Beck, *IJOC*, 2025]

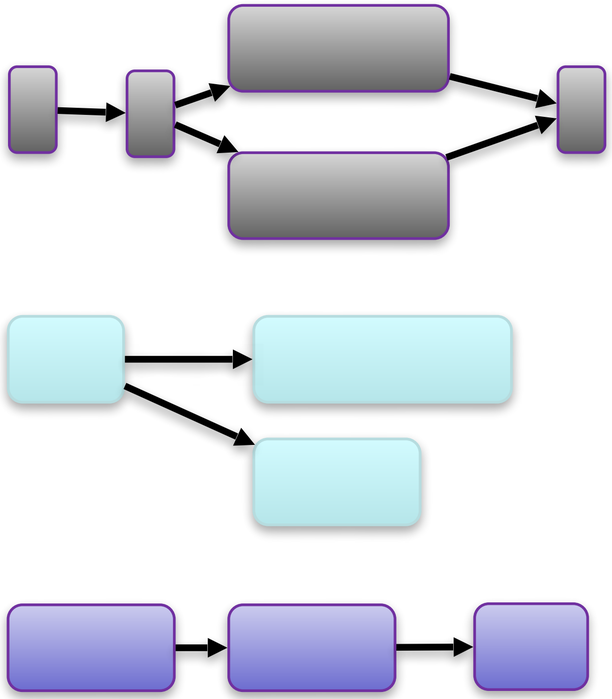


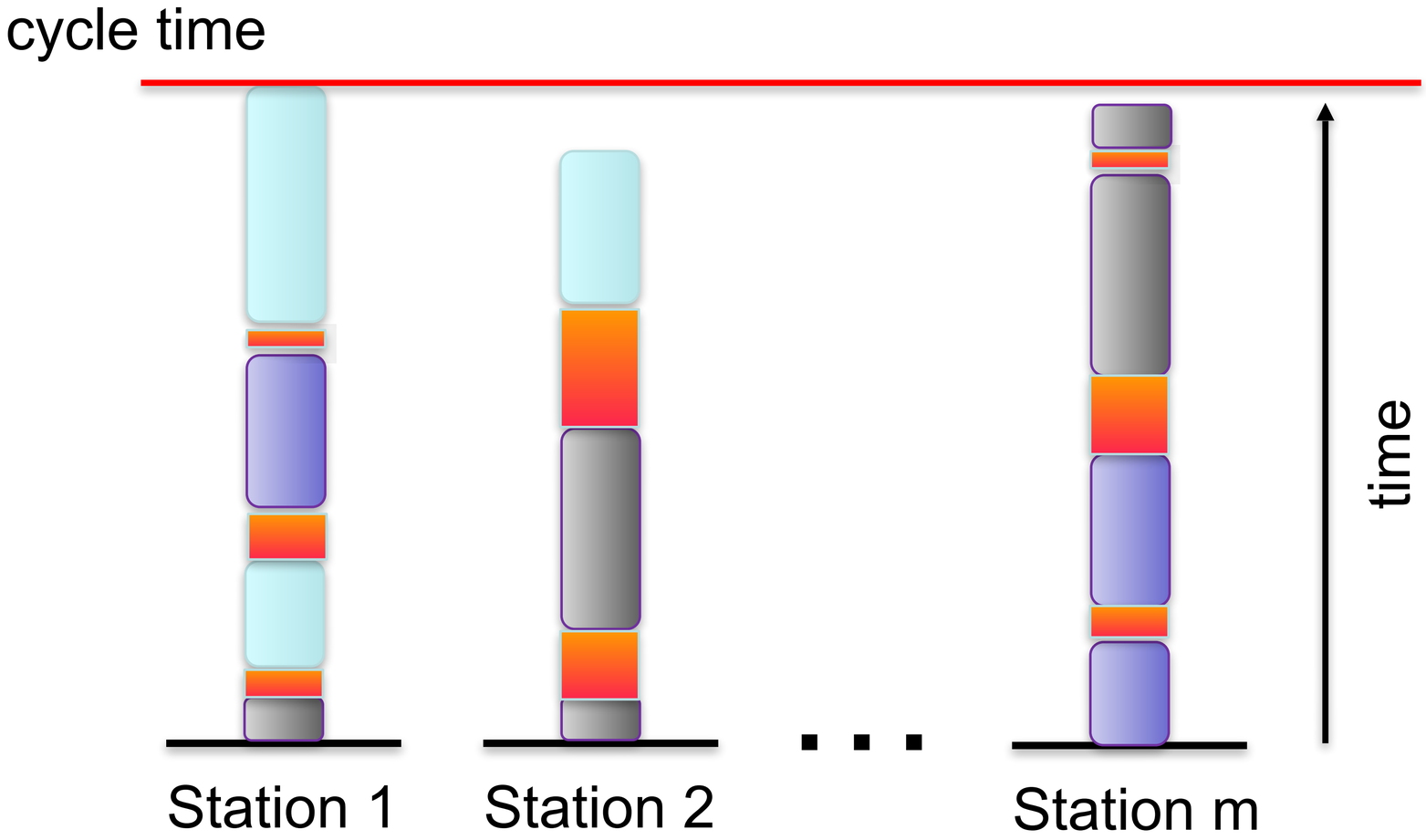


There is also a set-up between the last and first operations on a station ("backward" set-up)

P2: Setup Assembly Line Balancing

[Zhang & Beck, *IJOC*, 2025]





SUALBP-1: cycle time fixed, minimize m

SUALBP-2: m fixed, minimize cycle time

P2: MILP Model

[Esmaeilbeigi et al. , *OR Spectrum*, 2016]

eligible stations for operation i (preprocessing)

x_{ik}: Binary variable, $x_{ik} = 1$ iff task $i \in V$ is assigned to station $k \in FS_i$.

z_i: Integer variable representing the index of the station task $i \in V$ is assigned to.

u_k: Binary variable, $u_k = 1$ iff any task is assigned to station k.

g_{ijk}: Binary variable, $g_{ijk} = 1$ iff task i is performed immediately before task j on station k.

h_{ijk}: Binary variable, $h_{ijk} = 1$ iff task i is the last and j is the first task on station k.

r_i: Integer variable encoding the rank of task i in a sequence of all tasks.

P2: MILP Model

[Esmaeilbeigi et al. , *OR Spectrum*, 2016]

$$\min \sum_{k \in KP} u_k + \underline{m}$$

$$\text{s.t.} \sum_{k \in FS_i} x_{ik} = 1, \quad \forall i \in V,$$

$$\sum_{k \in FS_i} k \cdot x_{ik} = z_i, \quad \forall i \in V,$$

$$\sum_{i \in FT_k \cap F_i^F} g_{ijk} + \sum_{i \in FT_k \cap F_i^B} h_{ijk} = x_{ik}, \quad \forall i \in V,\ \forall k \in FS_i,$$

$$\sum_{i \in FT_k \cap P_j^F} g_{ijk} + \sum_{i \in FT_k \cap P_j^B} h_{ijk} = x_{jk}, \quad \forall j \in V,\ \forall k \in FS_j,$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = 1, \quad \forall k \in KD,$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = u_k, \quad \forall k \in KP,$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \cdot \left(\sum_{k \in (FS_i \cap FS_j)} g_{ijk} - 1 \right) \leq r_j, \quad \forall i \in V,\ \forall j \in F_i^F,$$

$$r_i + 1 \leq r_j, \quad \forall (i,j) \in \mathcal{E},$$

$$z_i \leq z_j, \quad \forall (i,j) \in \mathcal{E},$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} \cdot h_{ijk} \leq \overline{c}, \quad \forall k \in KD,$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} \cdot h_{ijk} \leq \overline{c} \cdot u_k, \quad \forall k \in KP,$$

$$\sum_{i \in FT_k \setminus \{j\}} x_{ik} \leq (n - \underline{m} + 1) \cdot (1 - h_{jjk}), \quad \forall k \in K,\ \forall j \in FT_k,$$

$$u_{k+1} \leq u_k, \quad \forall k \in KP \setminus \{\overline{m}\}.$$

$$g_{ijk} \in \{0,1\}, \quad \forall k \in K,\ \forall i \in FT_k,\ \forall j \in (FT_k \cap F_i^F),$$

$$h_{ijk} \in \{0,1\}, \quad \forall k \in K,\ \forall i \in FT_k,\ \forall j \in (FT_k \cap F_i^B),$$

$$|P_i^*| + 1 \leq r_i \leq n - |F_i^*|, \quad \forall i \in V,$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in V,\ \forall k \in FS_i,$$

$$r_i, z_i \in \mathbb{Z}^+, \quad \forall i \in V.$$

Pro-tip It is almost always a bad idea to present a complex model in a talk

P2: MILP Model

[Esmaeilbeigi et al. , *OR Spectrum*, 2016]

$$\min \sum_{k \in KP} u_k + \underline{m}$$

minimize # of stations ($\underline{m}$ lower bound, KP set of possible stations)

$$\text{s.t.} \sum_{k \in FS_i} x_{ik} = 1, \quad \forall i \in V,$$

each operation assigned to one station

$$\sum_{k \in FS_i} k \cdot x_{ik} = z_i, \quad \forall i \in V,$$

link index of machine assigned to i with binary assignment variable

$$\sum_{i \in FT_k \cap F_i^F} g_{ijk} + \sum_{i \in FT_k \cap F_i^B} h_{ijk} = x_{ik}, \quad \forall i \in V,\ \forall k \in FS_i,$$

$$\sum_{i \in FT_k \cap P_j^F} g_{ijk} + \sum_{i \in FT_k \cap P_j^B} h_{ijk} = x_{jk}, \quad \forall j \in V,\ \forall k \in FS_j,$$

each operation precedes and follows one other operation (including last → first)

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = 1, \quad \forall k \in KD,$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = u_k, \quad \forall k \in KP,$$

one last → first setup per station

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \cdot \left(\sum_{k \in (FS_i \cap FS_j)} g_{ijk} - 1 \right) \leq r_j, \quad \forall i \in V,\ \forall j \in F_i^F,$$

precedence constraints on station (partial)

P2: MILP Model

[Esmaeilbeigi et al. , *OR Spectrum*, 2016]

$$r_i + 1 \leq r_j, \quad \forall (i,j) \in \mathcal{E},$$

$$z_i \leq z_j, \quad \forall (i,j) \in \mathcal{E},$$

more precedence constraints

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} \cdot h_{ijk} \leq \bar{c}, \quad \forall k \in KD,$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} \cdot h_{ijk} \leq \bar{c} \cdot u_k, \quad \forall k \in KP,$$

station capacity constraints

$$\sum_{i \in FT_k \setminus \{j\}} x_{ik} \leq (n - \underline{m} + 1) \cdot (1 - h_{jjk}), \quad \forall k \in K, \ \forall j \in FT_k,$$

$$u_{k+1} \leq u_k, \quad \forall k \in KP \setminus \{\overline{m}\}.$$

all empty stations at the end

$$g_{ijk} \in \{0,1\}, \quad \forall k \in K, \ \forall i \in FT_k, \ \forall j \in (FT_k \cap F_i^F),$$

$$h_{ijk} \in \{0,1\}, \quad \forall k \in K, \ \forall i \in FT_k, \ \forall j \in (FT_k \cap F_i^B),$$

variable domains

$$|P_i^*| + 1 \leq r_i \leq n - |F_i^*|, \quad \forall i \in V,$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in V, \ \forall k \in FS_i,$$

$$r_i, z_i \in \mathbb{Z}^+, \quad \forall i \in V.$$

P2: CP Model

[Zhang & Beck, *IJOC*, 2025]

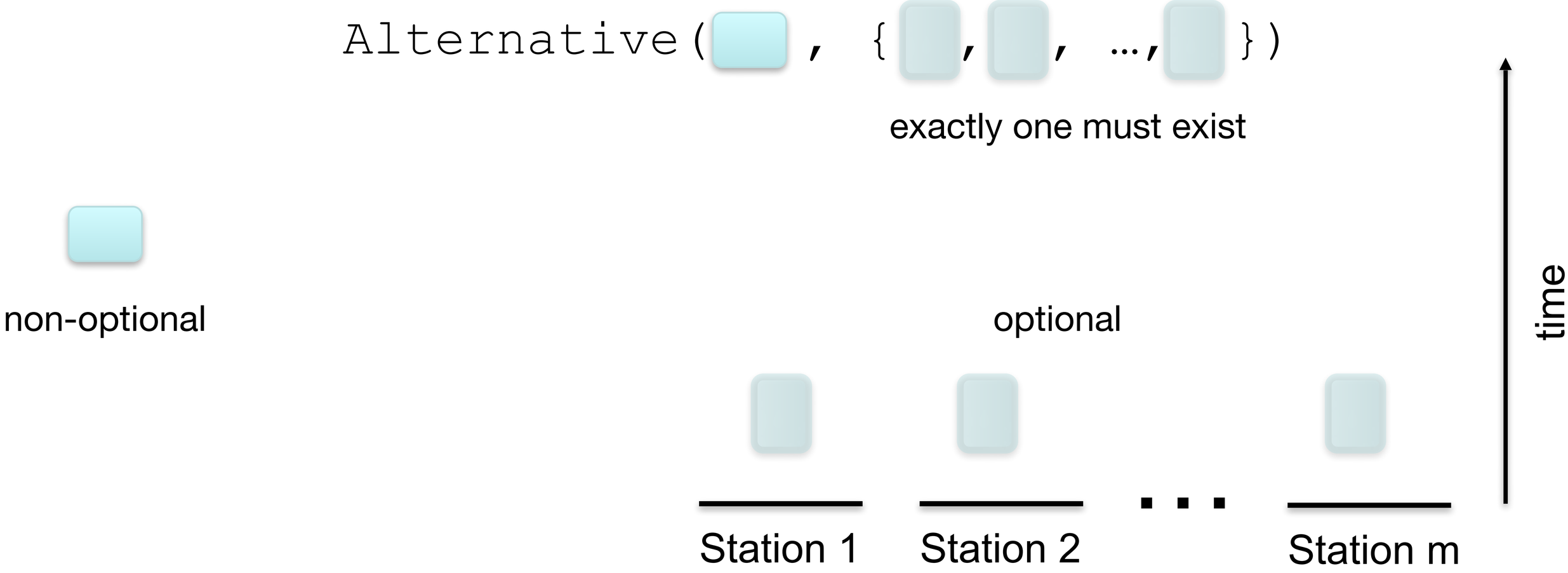
each operation is represented by $|FS_i| + 1$ optional interval variables

$task_i$ $(\overline{task}_i^k)$	(Optional) interval variable (Laborie et al. 2018) of task i (on station k $k \in FS_i$), with a size of t_i and a range between zero and c
d_s^k (d_t^k)	Interval variables of dummy first (last) task on station k, with size zero
$d_s^{k'}$	The next task after the dummy first task on station k
$d_t^{k'}$	The previous task before the dummy last task on station k

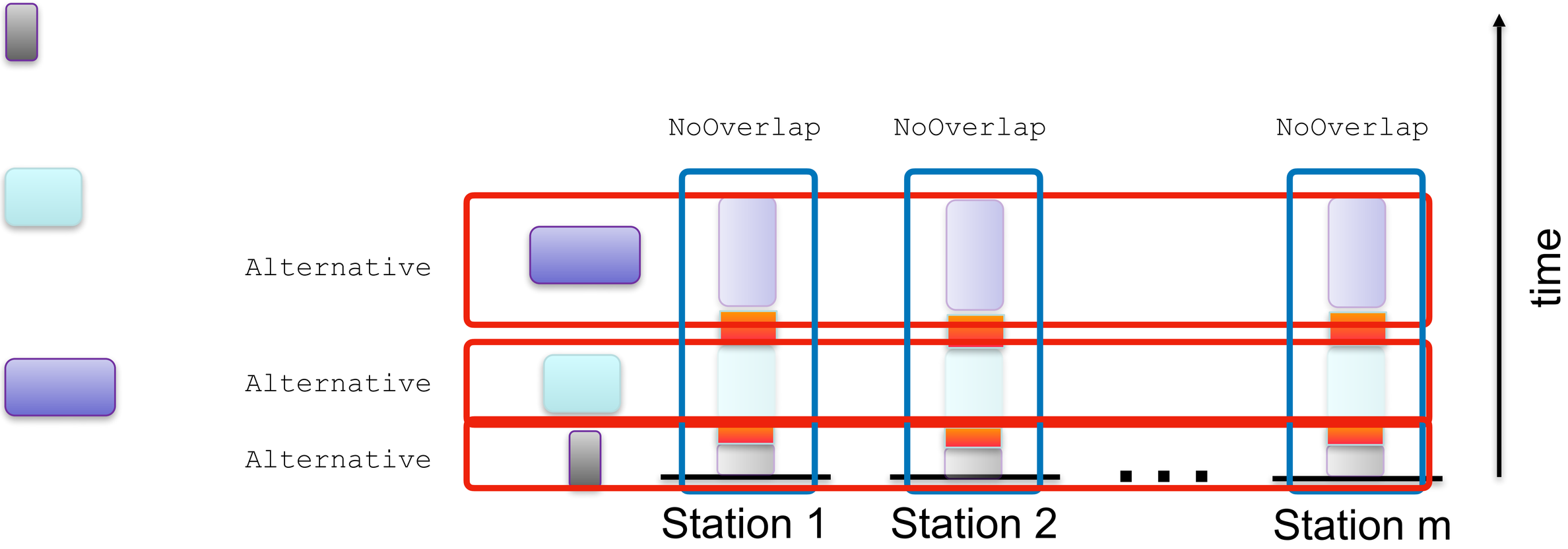
variables to keep track of first and last operations on a machine

u_k is a binary variable; $u_k = 1$ iff any task is assigned to station k.

P2: Setup Assembly Line Balancing



P2: Setup Assembly Line Balancing



P2: CP Model

[Zhang & Beck, *IJOC*, 2025]

$$\min \sum_{k \in KP} u_k + \underline{m}$$

s.t. $\texttt{Alternative}(task_i, \overline{task}_i^{E_i}, \ldots, \overline{task}_i^{L_i}), \quad \forall i \in V,$

$\texttt{NoOverlap}(\{\overline{task}_i^k \cup \{d_s^k, d_t^k\},\ \forall i \in FT_k\}, \{\tau_{ij},\ \forall i,j \in FT_k\}), \quad \forall k \in KD \cup KP,$

$\texttt{First}(\{\overline{task}_i^k,\ \forall i \in FT_k\} \cup \{d_s^k, d_t^k\}, d_s^k), \quad \forall k \in KD \cup KP,$

$\texttt{Last}(\{\overline{task}_i^k,\ \forall i \in FT_k\} \cup \{d_s^k, d_t^k\}, d_t^k), \quad \forall k \in KD \cup KP,$

$\texttt{PresenceOf}(\overline{task}_i^k) \leq 1, \quad \forall k \in KD,\ \forall i \in FT_k,$

$\texttt{PresenceOf}(\overline{task}_i^k) \leq u_k, \quad \forall k \in KP,\ \forall i \in FT_k,$

$\texttt{TypeOfNext}(d_s^k) = d_s^{k'}, \quad \forall k \in KD \cup KP,$

$\texttt{TypeOfPrev}(d_t^k) = d_t^{k'}, \quad \forall k \in KD \cup KP,$

$\texttt{StartOf}(d_t^k) + \texttt{Element}(\mu_{d_t^{k'} d_s^{k'}}) \leq c, \quad \forall k \in KD,$

$\texttt{StartOf}(d_t^k) + \texttt{Element}(\mu_{d_t^{k'} d_s^{k'}}) \leq c \cdot u_k, \quad \forall k \in KP,$

$\texttt{EndBeforeStart}(\overline{task}_i^k, \overline{task}_j^k), \quad \forall k \in KD \cup KP,\ \forall i \in FT_k,\ \forall j \in FT_k \cap F_i^*,$

$$\sum_{k \in FS_i} k \cdot \texttt{PresenceOf}(\overline{task}_i^k) \leq \sum_{k \in FS_j} k \cdot \texttt{PresenceOf}(\overline{task}_j^k), \quad \forall i \in V,\ \forall j \in F_i^*,$$

$$0 \leq task_i \leq c, \quad \forall i \in V,$$

$$0 \leq \overline{task}_i^k \leq c, \quad \forall i \in V,\ \forall k \in FS_i,$$

$$0 \leq d_s^k, d_t^k \leq c, \quad \forall k \in FS_i,$$

$$u_k = 1, \quad \forall k \in KD,$$

$$u_k \in \{0,1\}, \quad \forall k \in KP.$$

P2: CP Model

[Zhang & Beck, *IJOC*, 2025]

$$\min \sum_{k \in KP} u_k + \underline{m}$$

s.t. $\texttt{Alternative}(task_i, \overline{task}_i^{E_i}, \ldots, \overline{task}_i^{L_i}), \quad \forall i \in V,$

each operation is present on one station

$\texttt{NoOverlap}(\{\overline{task}_i^k \cup \{d_s^k, d_t^k\},\ \forall i \in FT_k\}, \{\tau_{ij},\ \forall i,j \in FT_k\}), \quad \forall k \in KD \cup KP,$

operations sequenced including set-up time

$\texttt{First}(\{\overline{task}_i^k,\ \forall i \in FT_k\} \cup \{d_s^k, d_t^k\}, d_s^k), \quad \forall k \in KD \cup KP,$

$\texttt{Last}(\{\overline{task}_i^k,\ \forall i \in FT_k\} \cup \{d_s^k, d_t^k\}, d_t^k), \quad \forall k \in KD \cup KP,$

specify first/last operations

$\texttt{PresenceOf}(\overline{task}_i^k) \leq 1, \quad \forall k \in KD,\ \forall i \in FT_k,$

$\texttt{PresenceOf}(\overline{task}_i^k) \leq u_k, \quad \forall k \in KP,\ \forall i \in FT_k,$

operation on station only if station is used

$\texttt{TypeOfNext}(d_s^k) = d_s^{k'}, \quad \forall k \in KD \cup KP,$

$\texttt{TypeOfPrev}(d_t^k) = d_t^{k'}, \quad \forall k \in KD \cup KP,$

assign first and last operations

$\texttt{StartOf}(d_t^k) + \texttt{Element}(\mu_{d_t^{k'} d_s^{k'}}) \leq c, \quad \forall k \in KD,$

$\texttt{StartOf}(d_t^k) + \texttt{Element}(\mu_{d_t^{k'} d_s^{k'}}) \leq c \cdot u_k, \quad \forall k \in KP,$

last → first set-up

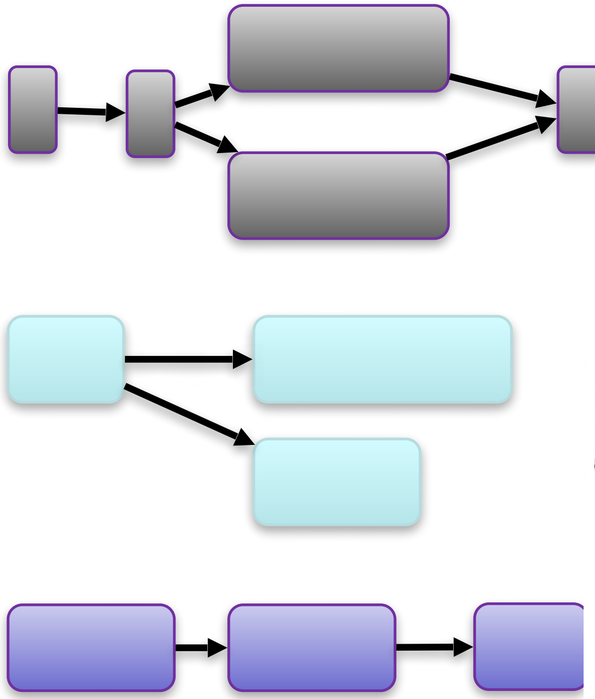
$\texttt{EndBeforeStart}(\overline{task}_i^k, \overline{task}_j^k), \quad \forall k \in KD \cup KP,\ \forall i \in FT_k,\ \forall j \in FT_k \cap F_i^*,$

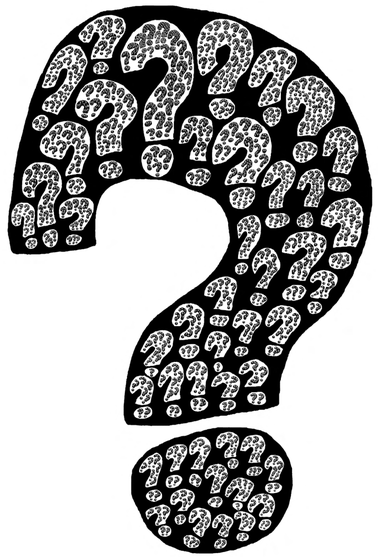
precedence on station

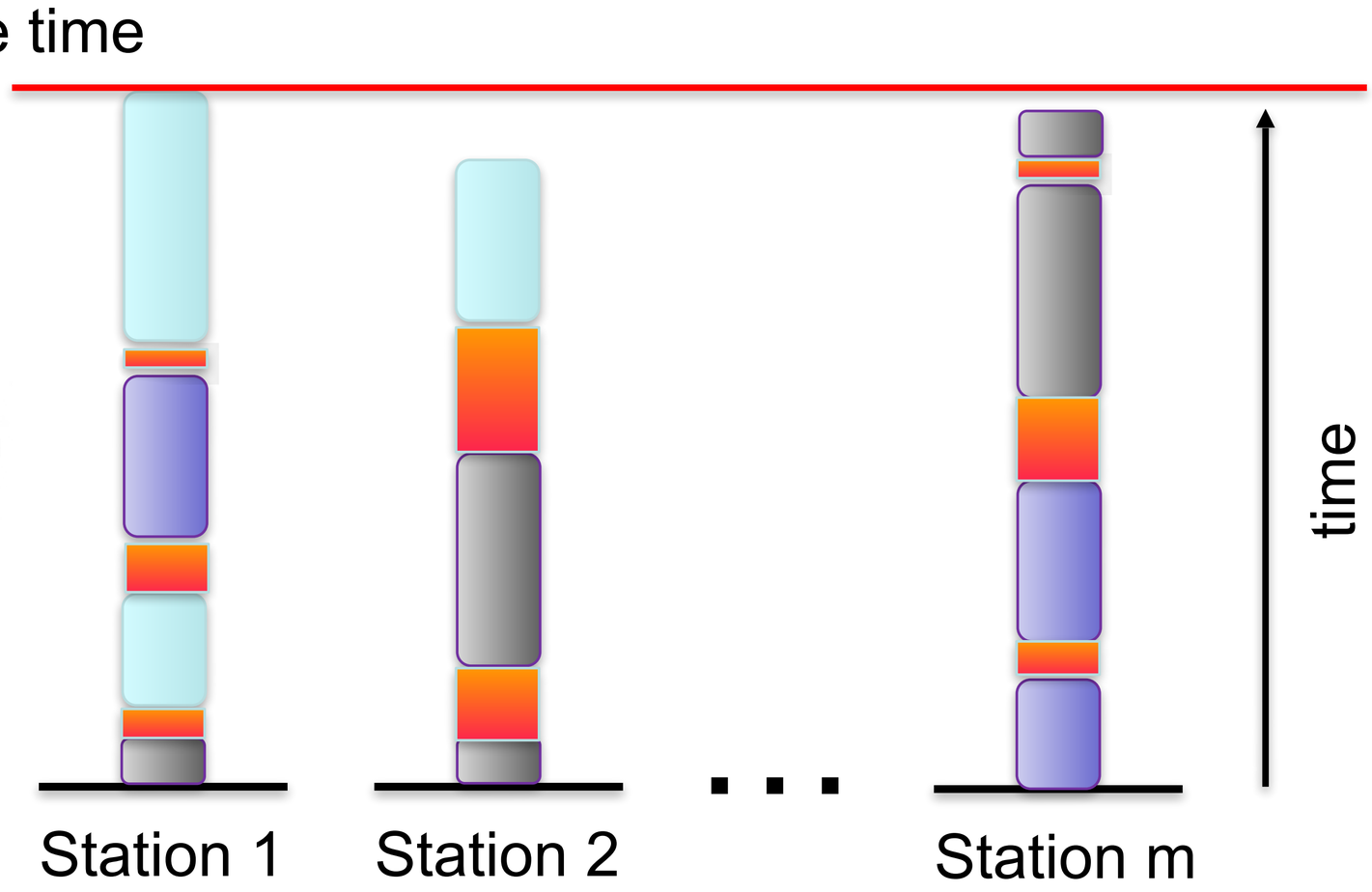
$\sum_{k \in FS_i} k \cdot \texttt{PresenceOf}(\overline{task}_i^k) \leq \sum_{k \in FS_j} k \cdot \texttt{PresenceOf}(\overline{task}_j^k), \quad \forall i \in V,\ \forall j \in F_i^*,$

precedence across stations

P2: Setup Assembly Line Balancing







So far: MILP and CP models

SUALBP-1: cycle time fixed, minimize m

SUALBP-2: m fixed, minimize cycle time

P2: DIDP Model

- Simple model
  - just pick an operation to go at the end of the current station
  - if nothing will fit, open a new station

P2: DIDP Model

[Zhang & Beck, *IJOC*, 2025]

State Variables

$$V(U, \kappa, p, f, r)$$

U: set of unassigned operations
κ: current station
p: previous task on current station
f: first task on current station
r: remaining time on current station

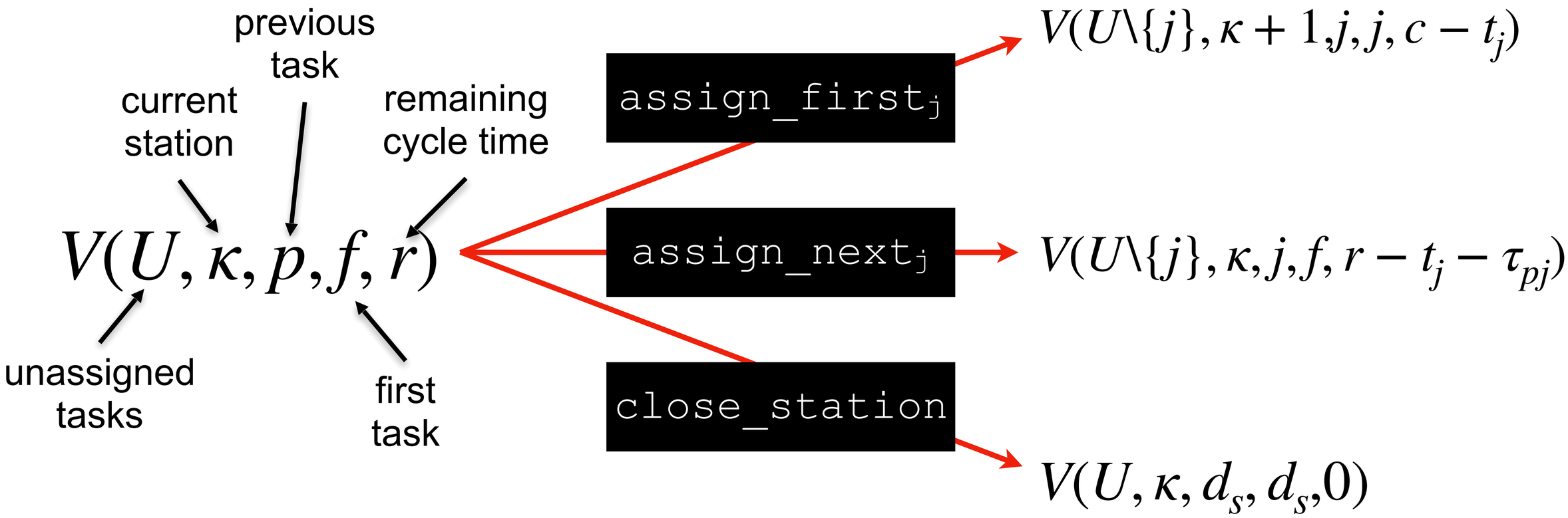
Starting state $\mathcal{V}(V, 0, d_s, d_s, 0)$

d_s: dummy task

P2: DIDP Model

[Zhang & Beck, *IJOC*, 2025]

Transitions



P2: DIDP Model

[Zhang & Beck, *IJOC*, 2025]

Bounds

Dominance

$$\mathcal{V}(U,\kappa,p,f,r) \leq \mathcal{V}(U,\kappa',p,f,r), \quad \text{if } \kappa \leq \kappa'$$

fewer stations better

$$\mathcal{V}(U,\kappa,p,f,r) \leq \mathcal{V}(U,\kappa,p,f,r'), \quad \text{if } r \geq r'$$

more remaining time better

Dual bounds

$$\mathcal{V}(U,\kappa,p,f,r) \geq$$

$$\max \begin{cases} \left\lceil \frac{\underline{\mu}_f + \sum_{i\in U}(\underline{\tau}_i + t_i) + (\overline{m}-\kappa)\cdot(\min_{i\in V}\underline{\mu}_i - \max_{i\in V}\underline{\tau}_i) - r}{c} \right\rceil, \\ \left\lceil \frac{\sum_{i\in U} t_i - r}{c} \right\rceil, \\ \sum_{i\in U} w_i^2 + \left\lceil \sum_{i\in U} w_i'^2 - l^2 \right\rceil, \\ \left\lceil \sum_{i\in U} w_i^3 - l^3 \right\rceil. \end{cases}$$

total remaining processing time divided by cycle time

mostly from bin-packing

P2: DIDP Model

[Zhang & Beck, *IJOC*, 2025]

compute $\mathcal{V}(V,0,d_s,d_s,0)$

$$\mathcal{V}(U,\kappa,p,f,r)$$

$$= \begin{cases} 0 & \text{if } U=\emptyset, f=d_s, \quad \text{base case} \\ 1+\min_{j\in U_1}\mathcal{V}(U\backslash\{j\},\kappa+1,j,j,c-t_j) & \text{if } U_1\neq\emptyset, f=d_s, \quad \text{assign\_first} \\ \min_{j\in U_2}\mathcal{V}(U\backslash\{j\},\kappa,j,f,r-t_j-\tau_{pj}) & \text{if } U_2\neq\emptyset, f\neq d_s, \quad \text{assign\_next} \\ \mathcal{V}(U,\kappa,d_s,d_s,0) & \text{if } U_3=\emptyset, f\neq d_s, \mu_{pf}\leq r, \quad \text{close\_station} \\ \infty & \text{otherwise,} \end{cases}$$

$$U_1=\{j\in U \mid U\cap P_j^*=\emptyset\},$$
$$U_2=\{j\in U_1 \mid r\geq t_j+\tau_{pi}\},$$
$$U_3=\{j\in U_1 \mid r\geq t_j+\tau_{pi}+\mu_{if}\}$$

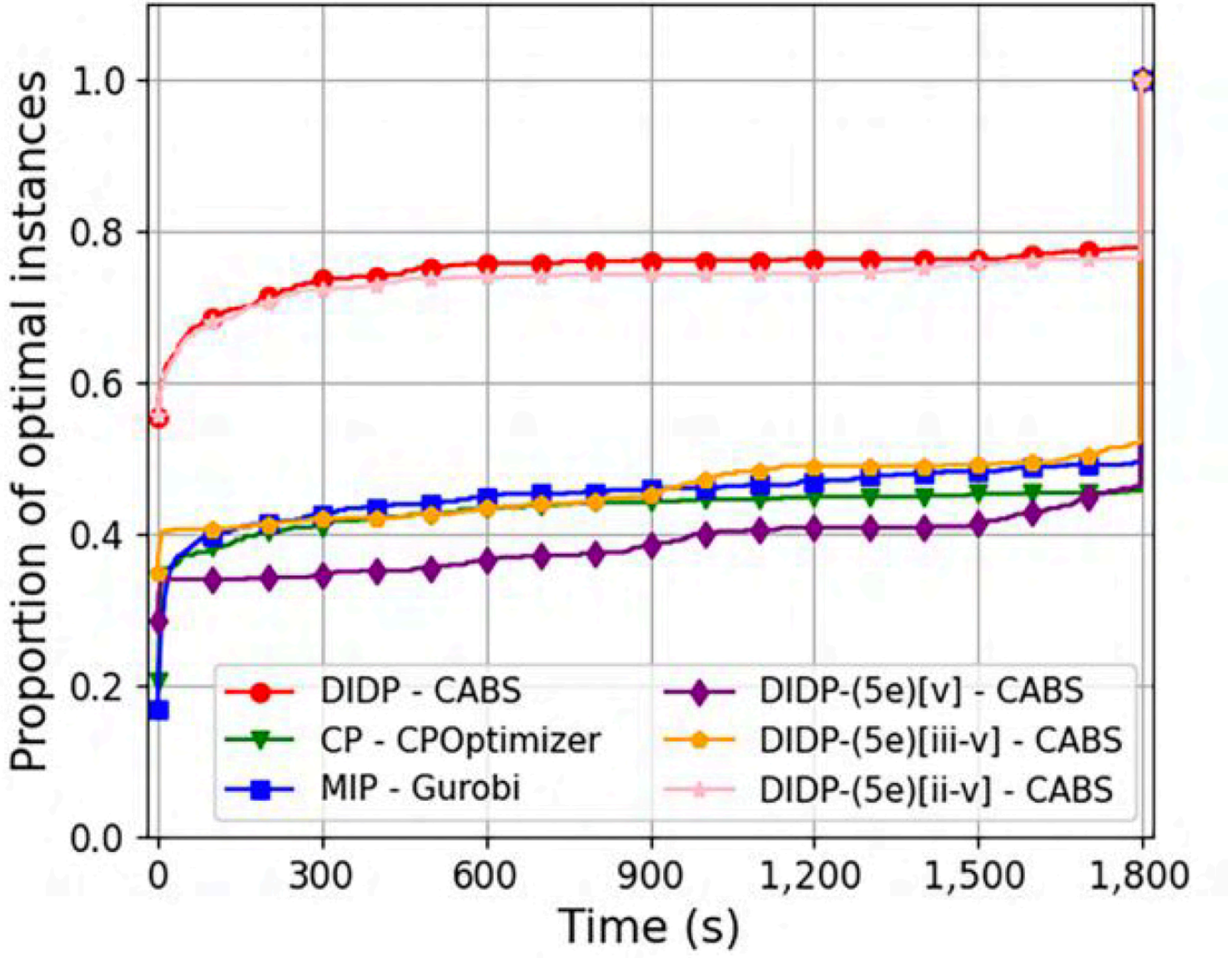
$$\mathcal{V}(U,\kappa,p,f,r)\leq\mathcal{V}(U,\kappa',p,f,r), \quad \text{if } \kappa\leq\kappa',$$

$$\mathcal{V}(U,\kappa,p,f,r)\leq\mathcal{V}(U,\kappa,p,f,r'), \quad \text{if } r\geq r',$$

$$\mathcal{V}(U,\kappa,p,f,r)\geq\max\begin{cases} \left\lceil \frac{\underline{\mu}_f+\sum_{i\in U}(\underline{\tau}_i+t_i)-(\overline{m}-\kappa)\cdot(\max_{i\in U}\underline{\tau}_i)+\max(\underline{m}-\kappa,0)\cdot(\min_{i\in U}\underline{\mu}_i)-r}{c}\right\rceil, \\ \left\lceil \frac{\underline{\mu}_f+\sum_{i\in U}t_i-r}{c}\right\rceil, \\ \sum_{i\in U}w_i^2+\left\lceil\sum_{i\in U}w_i'^2-l^2\right\rceil, \\ \left\lceil\sum_{i\in U}w_i^3-l^3\right\rceil, \\ 0. \end{cases}$$

P2: Results SUALBP-1

[Zhang & Beck, *IJOC*, 2025]

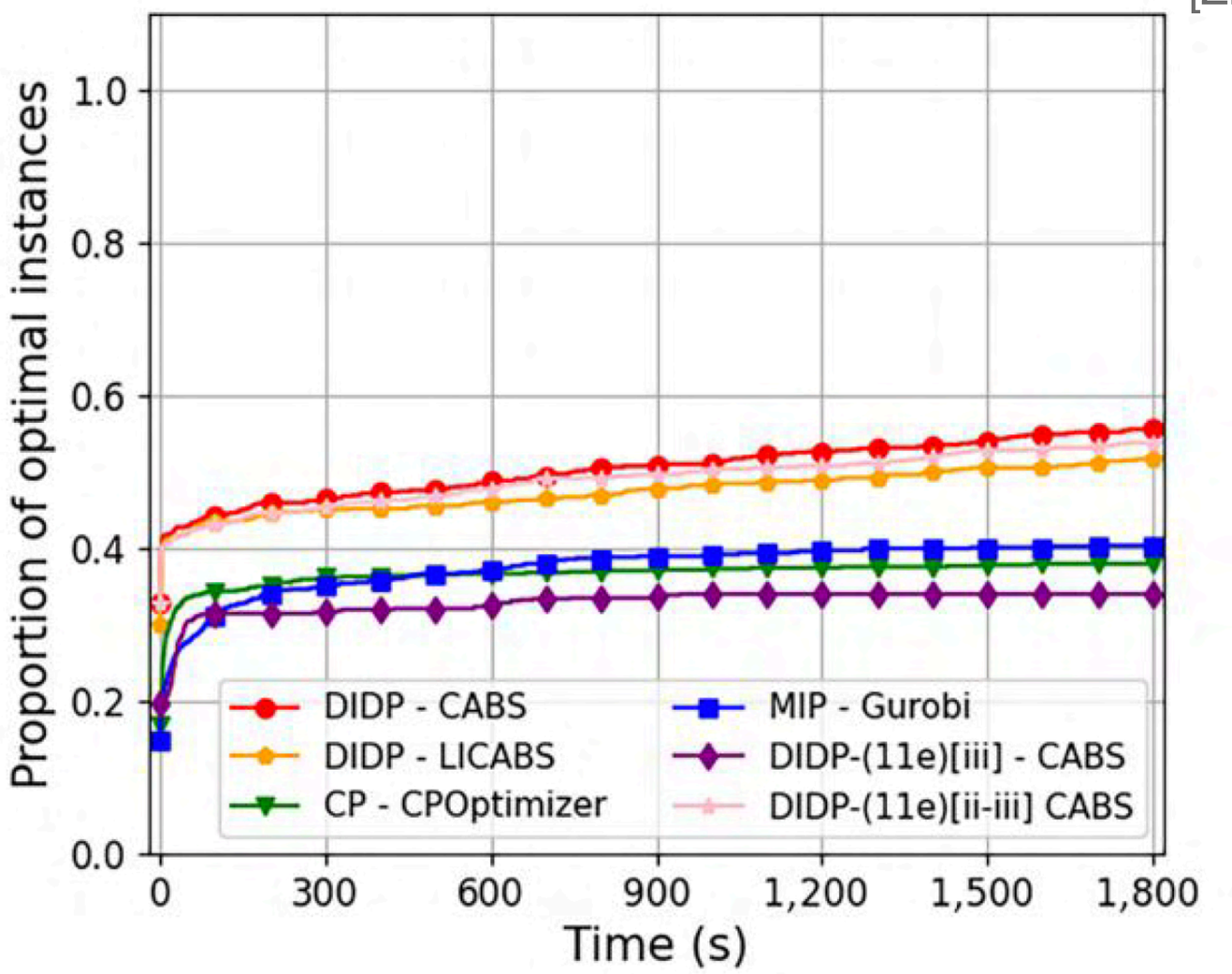


788 problem instances from the literature

P2: Results SUALBP-2

[Zhang & Beck, *IJOC*, 2025]

I have not presented the SUALBP-2 models



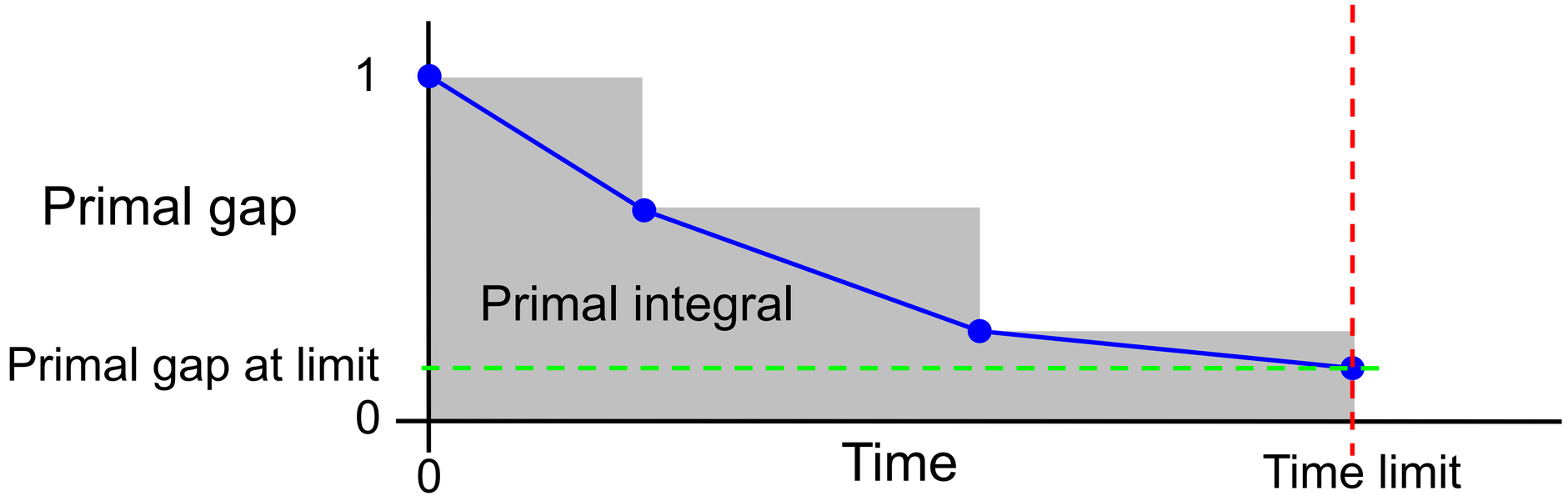
788 problem instances from the literature

SUALBP-1: cycle time fixed, minimize m
SUALBP-2: m fixed, minimize cycle time

Primal Integral

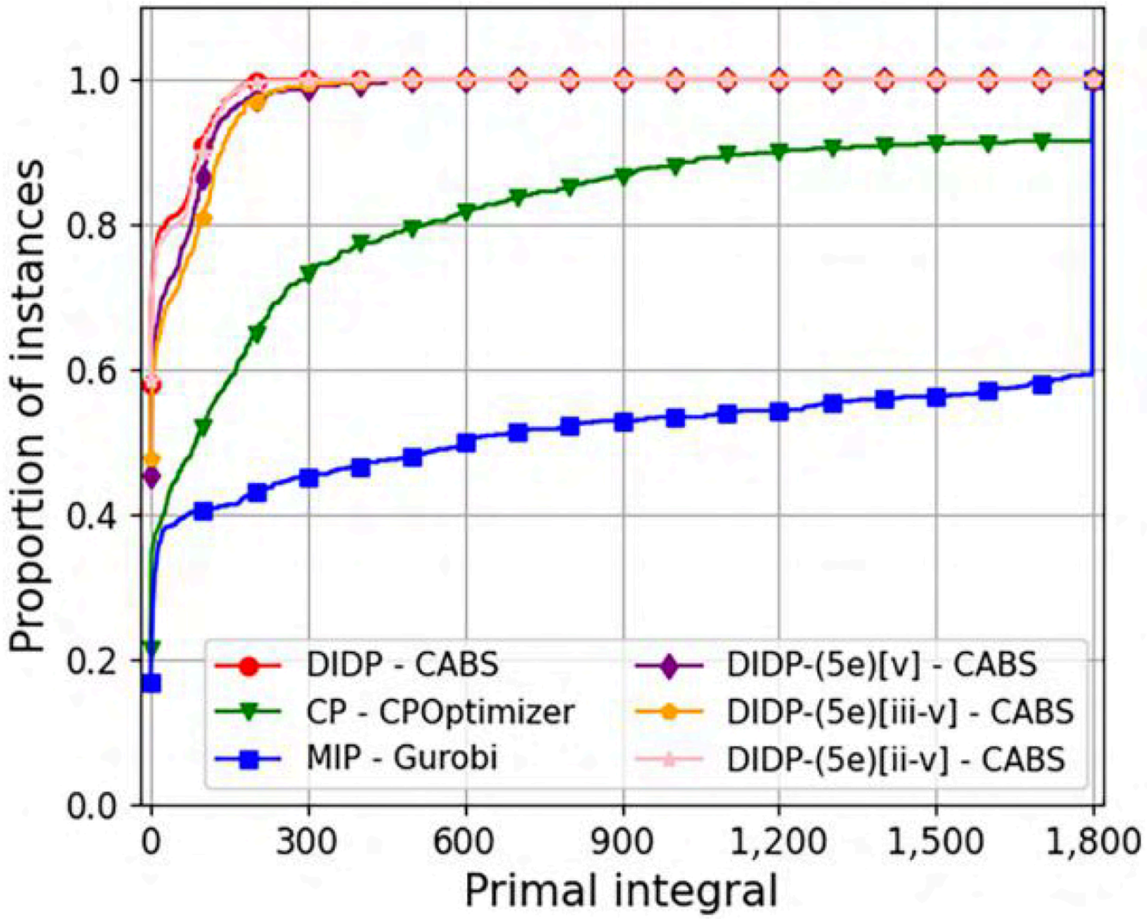
[Berthold, *OR Letters*, 2013]

- Primal gap: $\frac{\text{solution cost} - \text{best known cost}}{\text{solution cost}}$ (1 if no solution found)

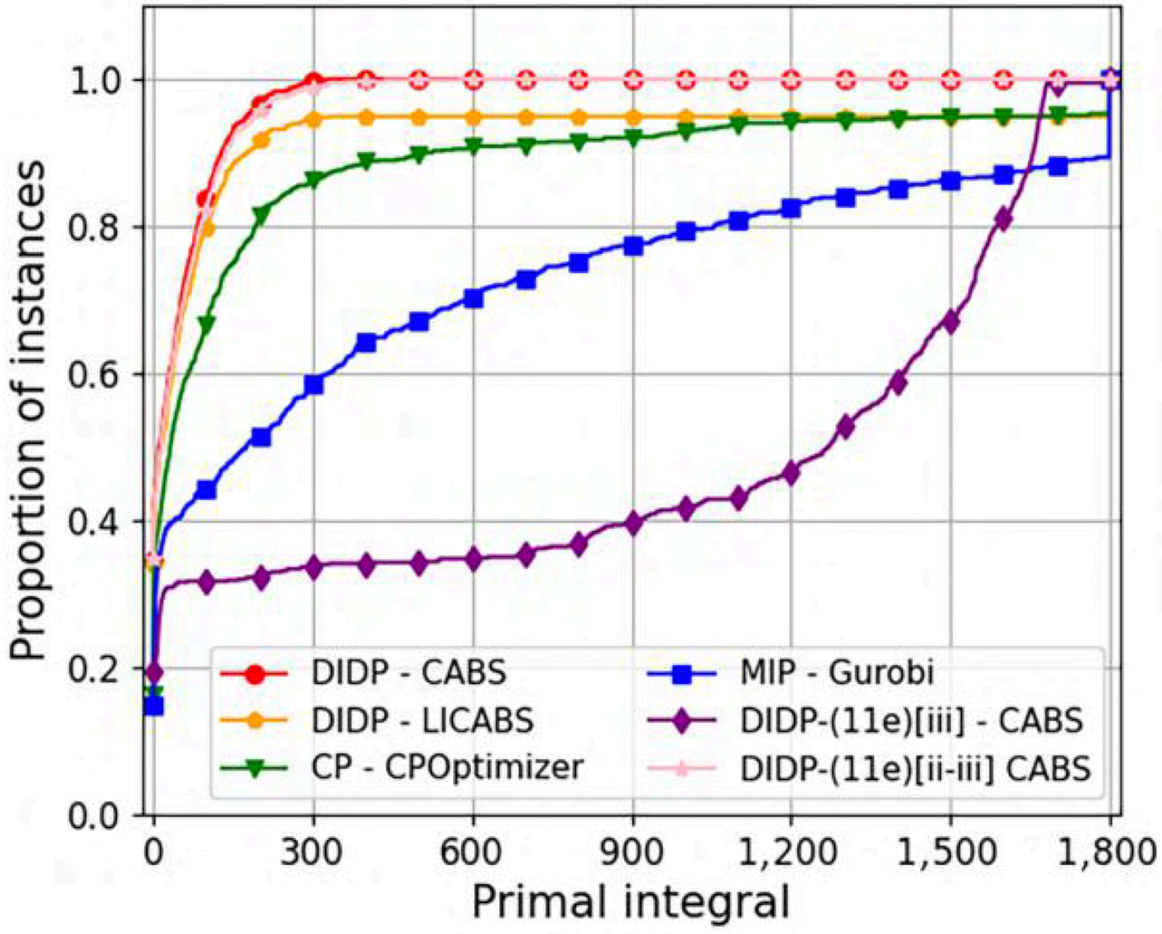


P2: Primal Integral Results

[Zhang & Beck, *IJOC*, 2025]



SUALBP-1



SUALBP-2

P2: Summary

- Fairly complex combination of assignment and sequencing
- DIDP model is best by a reasonable gap
  - then MILP then CP (depending on what you count)
- The modeling approaches used in all three models are generally applicable in scheduling

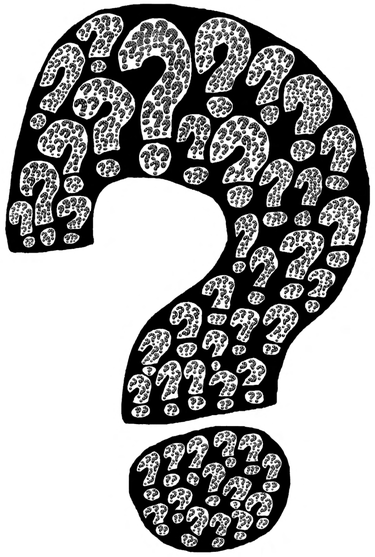
P2: Summary

MILP

- assignment and immediate-predecessor variables
- no explicit representation of start times

CP

- interval variables
- no-overlap
- alternative



DIDP

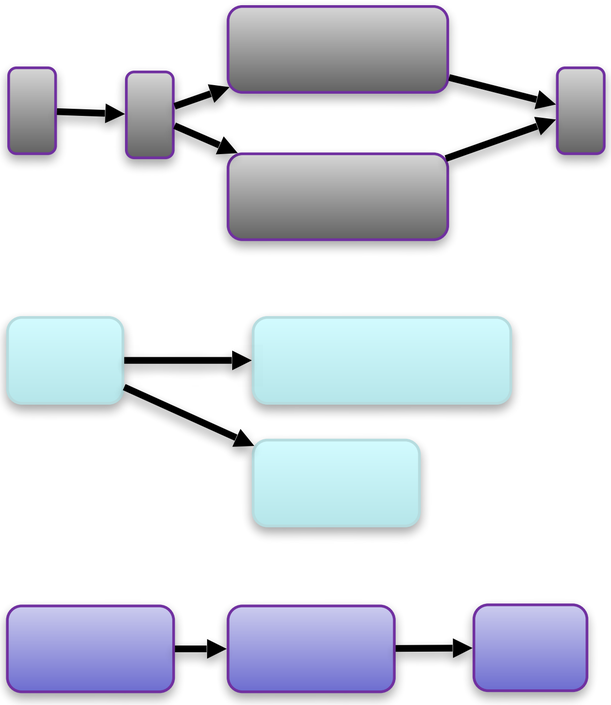
- dual bounds
- SUALBP-1 has a coarse objective function
- many optimal solutions

P3: Multi-mode Resource-Constrained Multi-Project Scheduling Problem (MRCMPSP) with global and local resources

Welcome to the next level

P3

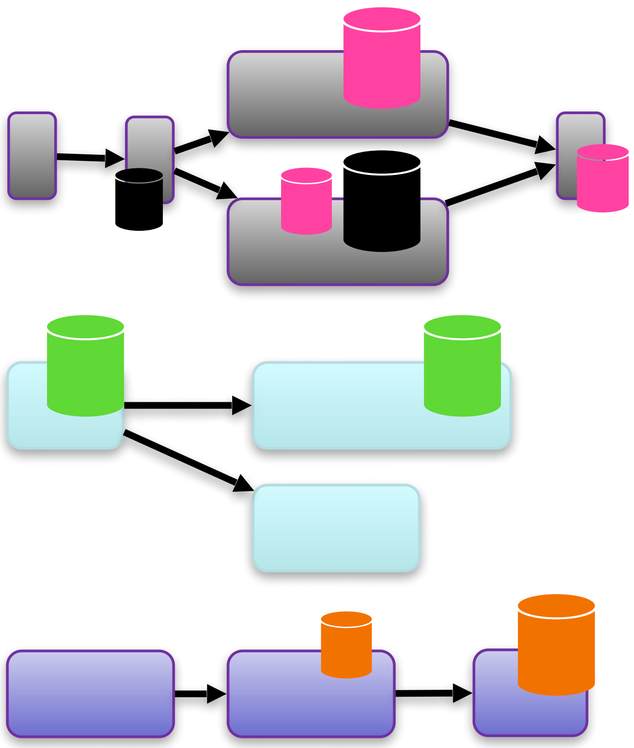
Multi-mode Resource-Constrained Multi-Project Scheduling with global and local resources



Project: a set of temporally (and logically) related operations

P3

Multi-mode Resource-Constrained Multi-Project Scheduling with global and local resources



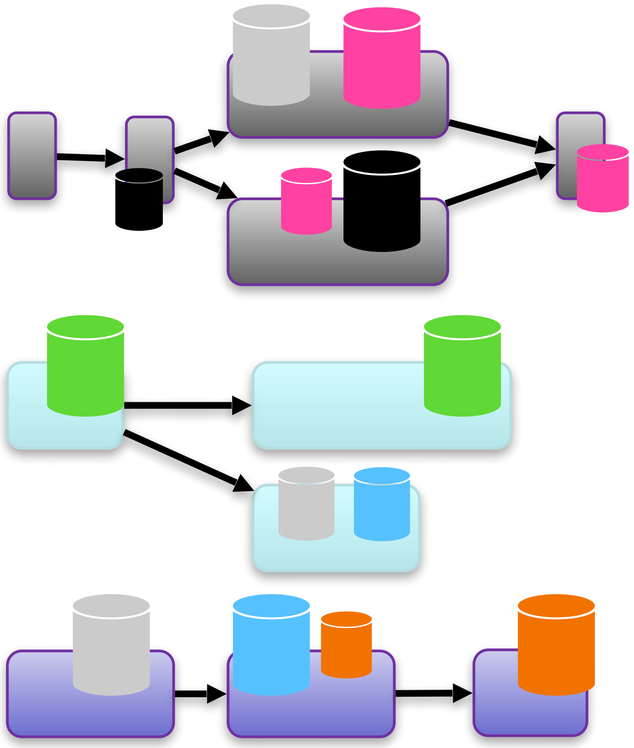
Project: a set of temporally (and logically) related operations

Resource-Constrained: each operation requires some units of some resource (possibly 0)

Resource: machine, worker, fuel, raw material, …

P3

Multi-mode Resource-Constrained Multi-Project Scheduling with global and local resources



Project: a set of temporally (and logically) related operations

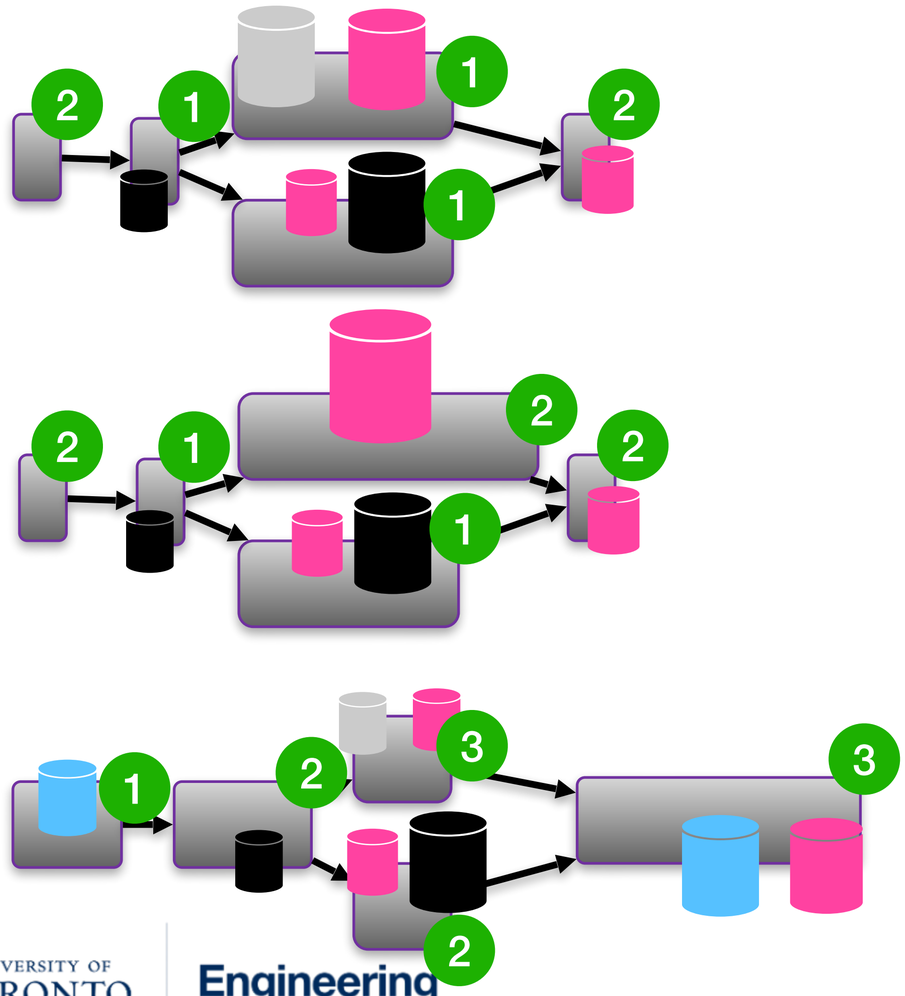
Resource-Constrained: each operation requires some units of some resource (possibly 0)

local resource: only required within one project

global resource: required across projects

P3

Multi-mode Resource-Constrained Multi-Project Scheduling with global and local resources



Mode: each operation has multiple “modes” of execution which change processing times and resource requirements

Representing resource/time trade-offs (e.g., 2 days with 3 people, 4 days with 2)

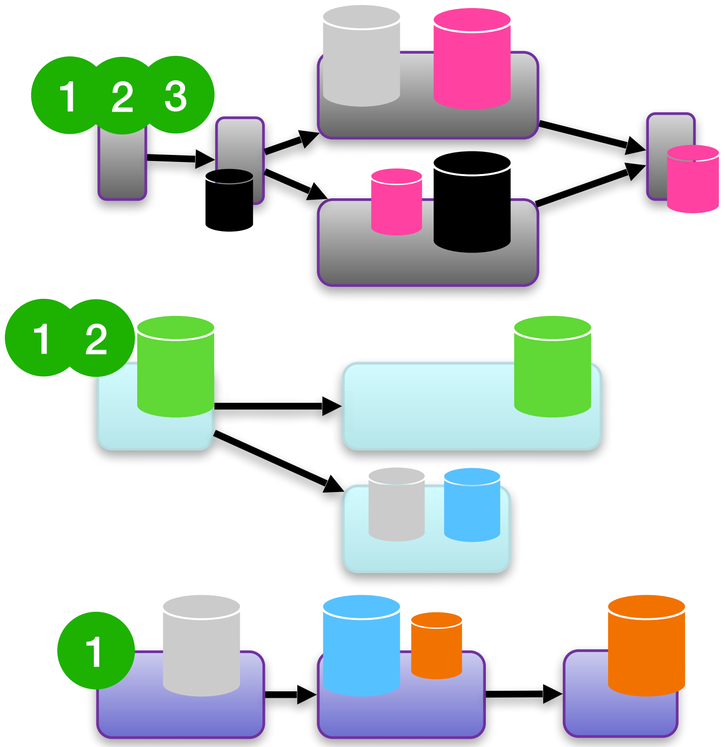
P3: Renewable vs. Non-renewable

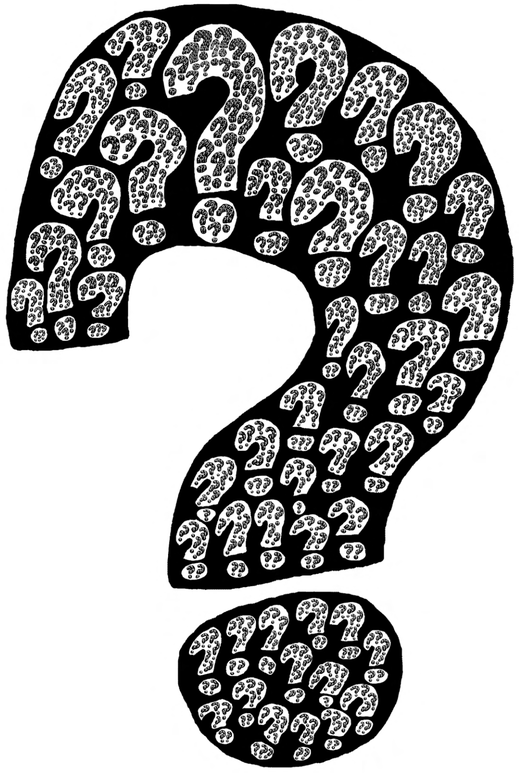
- Two types of resources
  - Renewable: fixed amount available per time unit
    - e.g., machines, people, …
    - used by an operation
  - Non-renewable: fixed amount available for the whole horizon
    - e.g., fuel, battery, raw materials, …
    - consumed by an operation

Global resources are renewable, local resources can be either renewable or non-renewable

P3: MRCMPSP

- Assign a mode and a start time to each operation to respect resource capacities, to satisfy precedence constraints, and to minimize sum of project makespans (“delay”)





P3: MILP Model

$x_{ijmt} \in \{0,1\}$: 1 iff operation j of project i is executed in mode m and starts at time t

P3: MILP Model

Minimize Project Delay

$$\min \sum_{i \in \mathcal{P}} \sum_{m \in \mathcal{M}_{ij}} \sum_{t \in \mathcal{T}} \left[t \cdot x_{i,n_i+1,mt} - (f_i + l_i) \right]$$

start of dummy sink node

constant for project i

Assignment & Precedence Constraints

$$s.t. \sum_{m \in \mathcal{M}_{ij}} \sum_{t \in \mathcal{T}_{ij}} x_{ijmt} = 1 \qquad \forall i \in \mathcal{P}, j \in \mathcal{V}_i$$

every operation starts once

$$\sum_{m \in \mathcal{M}_{ij}} \sum_{t \in \mathcal{T}_{ij}} (t + d_{ijm}) x_{ijmt} \leq \sum_{m \in \mathcal{M}_{ij'}} \sum_{t \in \mathcal{T}_{ij}} t x_{ij'mt} \quad \forall i \in \mathcal{P}, (j, j') \in \mathcal{E}_i$$

Global & Local Resources Constraints

$$\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \sum_{s=\max(ES_{ij}, t-d_{ijm}+1)}^{\min(LS_{ij}, t)} r^G_{ijmk} x_{ijms} \leq R^G_k \quad \forall k \in \mathcal{R}^G, t \in \mathcal{T}$$

time-indexed renewable resources

$$\sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \sum_{s=\max(ES_{ij}, t-d_{ijm}+1)}^{\min(LS_{ij}, t)} r^{LR}_{ijmk} x_{ijms} \leq R^{LR}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LR}_i, t \in \mathcal{T}$$

$$\sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \sum_{t \in \mathcal{T}_{ij}} r^{LN}_{ijmk} x_{ijmt} \leq R^{LN}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LN}_i$$

knapsack non-renewable resources

P3: CP Model

Interval variable for operation j of project i

$$y_{ij} = \texttt{IntervalVar}(\text{start} = [ES_{ij}, LS_{ij}]) \qquad \forall i \in \mathcal{P}, j \in \mathcal{V}_i$$

Optional interval variable for operation j of project i in mode m

$$\tilde{y}_{ijm} = \texttt{IntervalVar}(\text{length} = d_{ijm}, \text{optional} = \texttt{TRUE}) \; \forall i \in \mathcal{P}, j \in \mathcal{V}_i, m \in \mathcal{M}_{ij}$$

P3: CP Model

Minimize Project Delay

$$\min \sum_{i \in \mathcal{P}} \left[\texttt{StartOf}(y_{i,n_i+1}) - (f_i + l_i) \right]$$

Assignment & Precedence Constraints

$$s.t.\ \texttt{Alternative}(y_{ij}, \{\tilde{y}_{ijm} \forall m \in \mathcal{M}_{ij}\}) \qquad \forall i \in \mathcal{P},\ j \in \mathcal{V}_i$$

$$\texttt{EndBeforeStart}(y_{ij},\ y_{ij'}) \qquad \forall i \in \mathcal{P}, (j, j') \in \mathcal{E}_i$$

Global & Local Resources Constraints

$$\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \texttt{Pulse}(\tilde{y}_{ijm},\ r^G_{ijmk}) \leq R^G_k \ \forall k \in \mathcal{R}^G$$

cumulative functions

$$\sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \texttt{Pulse}(\tilde{y}_{ijm},\ r^{LR}_{ijmk}) \leq R^{LR}_{ik} \qquad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LR}_i$$

$$\sum_{j \in \mathcal{V}_i} \sum_{m \in \mathcal{M}_{ij}} \texttt{PresenceOf}(\tilde{y}_{ijm}) r^{LN}_{ijmk} \leq R^{LN}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LN}_i$$

P3: MILP and CP Models

Integer Program (IP)

Minimize Project Delay

$$\min \sum_{i\in\mathcal{P}} \sum_{m\in\mathcal{M}_{ij}} \sum_{t\in\mathcal{T}} \left[t \cdot x_{i,n_i+1,mt} - (f_i + l_i)\right] \quad (3.1)$$

Assignment & Precedence Constraints

$$s.t. \sum_{m\in\mathcal{M}_{ij}} \sum_{t\in\mathcal{T}_{ij}} x_{ijmt} = 1 \qquad \forall i \in \mathcal{P}, j \in \mathcal{V}_i \quad (3.2)$$

$$\sum_{m\in\mathcal{M}_{ij}} \sum_{t\in\mathcal{T}_{ij}} (t + d_{ijm}) x_{ijmt} \leq \sum_{m\in\mathcal{M}_{ij'}} \sum_{t\in\mathcal{T}_{ij}} t x_{ij'mt} \quad \forall i \in \mathcal{P}, (j, j') \in \mathcal{E}_i \quad (3.3)$$

Global & Local Resources Constraints

$$\sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \sum_{s=\max(ES_{ij}, t-d_{ijm}+1)}^{\min(LS_{ij},t)} r^G_{ijmk} x_{ijms} \leq R^G_k \quad \forall k \in \mathcal{R}^G, t \in \mathcal{T} \quad (3.4)$$

$$\sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \sum_{s=\max(ES_{ij}, t-d_{ijm}+1)}^{\min(LS_{ij},t)} r^{LR}_{ijmk} x_{ijms} \leq R^{LR}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LR}_i, t \in \mathcal{T} \quad (3.5)$$

$$\sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \sum_{t\in\mathcal{T}_{ij}} r^{LN}_{ijmk} x_{ijmt} \leq R^{LN}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LN}_i \quad (3.6)$$

Variable Domains

$$x_{ijmt} \in \{0, 1\} \quad \forall i \in \mathcal{P}, j \in \mathcal{V}_i, m \in \mathcal{M}_{ij}, t \in \mathcal{T}_{ij} \quad (3.7)$$

➕ Linear relaxation naturally give lower bounds

➖ Scales poorly

Constraint Program (CP)

Minimize Project Delay

$$\min \sum_{i\in\mathcal{P}} \left[\texttt{StartOf}(y_{i,n_i+1}) - (f_i + l_i)\right] \quad (3.8)$$

Assignment & Precedence Constraints

$$s.t.\ \texttt{Alternative}(y_{ij}, \{\tilde{y}_{ijm} \forall m \in \mathcal{M}_{ij}\}) \qquad \forall i \in \mathcal{P},\ j \in \mathcal{V}_i \quad (3.9)$$

$$\texttt{EndBeforeStart}(y_{ij},\ y_{ij'}) \qquad \forall i \in \mathcal{P}, (j, j') \in \mathcal{E}_i \quad (3.10)$$

Global & Local Resources Constraints

$$\sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \texttt{Pulse}(\tilde{y}_{ijm},\ r^G_{ijmk}) \leq R^G_k \ \forall k \in \mathcal{R}^G \quad (3.11)$$

$$\sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \texttt{Pulse}(\tilde{y}_{ijm},\ r^{LR}_{ijmk}) \leq R^{LR}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LR}_i \quad (3.12)$$

$$\sum_{j\in\mathcal{V}_i} \sum_{m\in\mathcal{M}_{ij}} \texttt{PresenceOf}(\tilde{y}_{ijm}) r^{LN}_{ijmk} \leq R^{LN}_{ik} \quad \forall i \in \mathcal{P}, k \in \mathcal{R}^{LN}_i \quad (3.13)$$

Variable Domains

$$y_{ij} = \texttt{IntervalVar}(\text{start}=[ES_{ij}, LS_{ij}]) \qquad \forall i \in \mathcal{P}, j \in \mathcal{V}_i \quad (3.14)$$

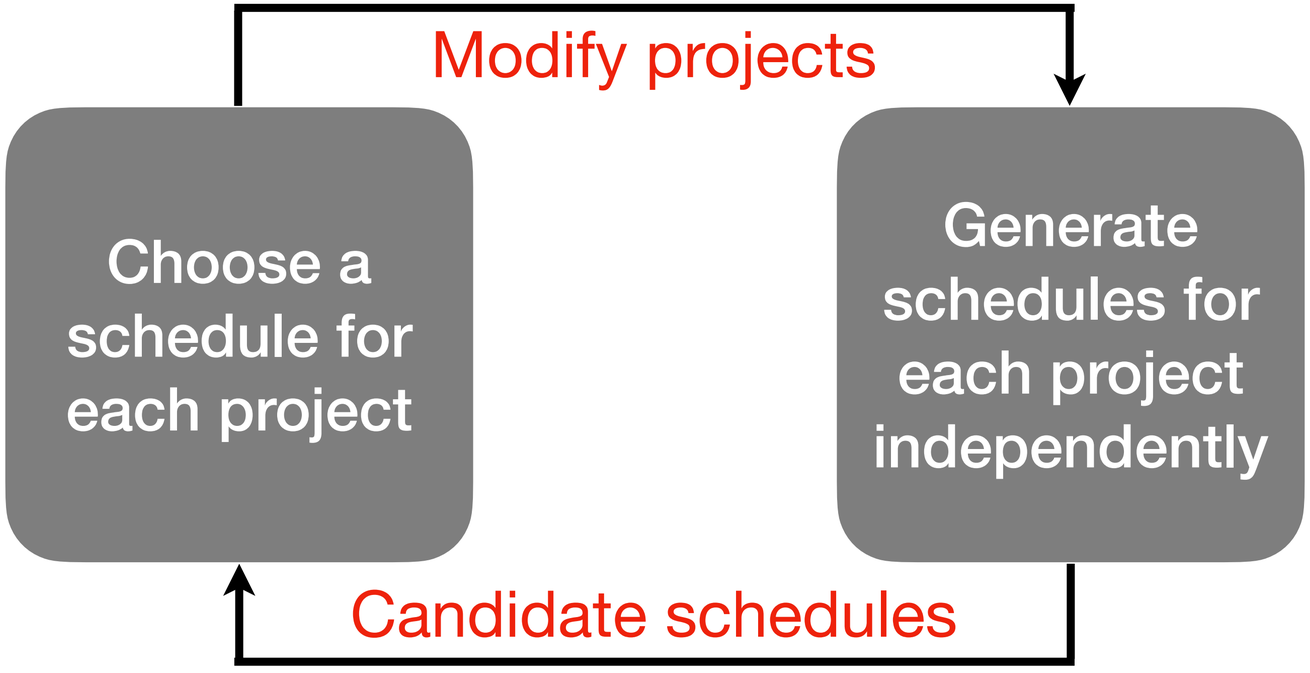
$$\tilde{y}_{ijm} = \texttt{IntervalVar}(\text{length}=d_{ijm}, \text{optional}=\texttt{TRUE}) \quad \forall i \in \mathcal{P}, j \in \mathcal{V}_i, m \in \mathcal{M}_{ij} \quad (3.15)$$

➕ Scales well

➖ No effective lower bounding mechanism

P3: Decomposition

- Observation: if we ignore the global resources, each project is an independent problem
- Can we exploit this?



⚠️Warning⚠️

- We’re taking a bit of a detour to cover a general solution approach in MILP
- This is an advanced MILP approach with a lot of beautiful math underneath it (which I will skip)
- If this is your first time seeing it, it may look a bit like magic

Goal
at least remember the idea and the names:
Dantzig-Wolfe Decomposition, Column Generation, Branch-and-Price

MILPs and Matrices

- MILPs can be represented in matrix form

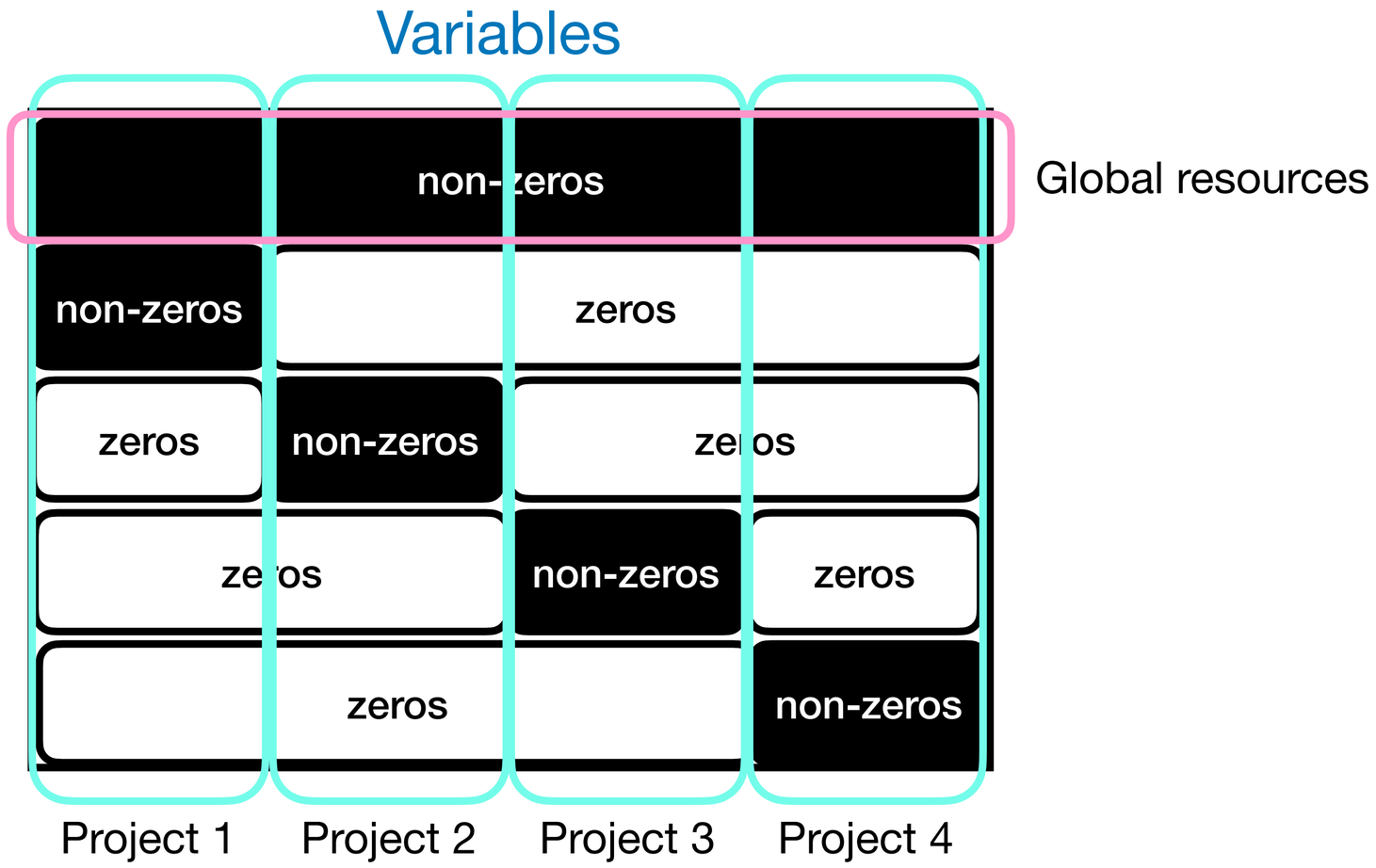
$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2,n-1} & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-1,1} & a_{n-1,2} & \cdots & a_{m-1,n-1} & a_{m-1,n} \\ a_{m1} & a_{m2} & \cdots & a_{n,n-1} & a_{mn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}$$

Dantzig-Wolfe Decompostion

- Our observation is a general pattern that can occur

A matrix representing a MILP

Linear Constraints



Dantzig-Wolfe Master Problem

$$\min \sum_{i \in \mathcal{P}} \sum_{\omega \in \Omega_i} c_{i\omega} \lambda_{i\omega}$$

minimize sum of makespans

$$s.t. \sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1$$

each project has one schedule $\forall i \in \mathcal{P}$

$$\sum_{i \in \mathcal{P}} \sum_{\omega \in \Omega_i} r_{i\omega kt} \lambda_{i\omega} \leq R_k^G$$

resource constraints $\forall k \in \mathcal{R}^G,\ t \in \mathcal{T}$

$$\lambda_{i\omega} \in \{0, 1\} \qquad \forall i \in \mathcal{P},\ \omega \in \Omega_i$$

Ω_i: set of all schedules for project i

$c_{i\omega}$: makespan of schedule $\omega \in \Omega_i$

$r_{i\omega kt}$: usage of global resource k at time t by schedule $\omega \in \Omega_i$

Big Problem:
Ω_i has exponential size

Solution: Generate Ω_i Lazily

1. Start with some project schedules $\tilde{\Omega}_i \subseteq \Omega_i$
2. Solve Restricted Master Problem (RMP)
3. Use RMP solution to generate new project schedules
4. Add new project schedules to $\tilde{\Omega}_i$
5. Goto 2

This does not quite work

But it does work to solve the linear relaxation of the RMP

Dantzig-Wolfe Master Problem

$$\min \sum_{i \in \mathcal{P}} \sum_{\omega \in \Omega_i} c_{i\omega} \lambda_{i\omega}$$

minimize sum of makespans

$$s.t. \sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1$$

each project has one schedule $\forall i \in \mathcal{P}$

$$\sum_{i \in \mathcal{P}} \sum_{\omega \in \Omega_i} r_{i\omega kt} \lambda_{i\omega} \leq R_k^G$$

resource constraints $\forall k \in \mathcal{R}^G,\ t \in \mathcal{T}$

$$\lambda_{i\omega} \in \{0, 1\} \qquad \forall i \in \mathcal{P},\ \omega \in \Omega_i$$

Ω_i: set of all schedules for project i

$c_{i\omega}$: makespan of schedule $\omega \in \Omega_i$

$r_{i\omega kt}$: usage of global resource k at time t by schedule $\omega \in \Omega_i$

Dantzig-Wolfe Restricted Master Problem

$\tilde{\Omega}_i$: set of some schedules for project i

$c_{i\omega}$: makespan of schedule $\omega \in \tilde{\Omega}_i$

$r_{i\omega kt}$: usage of global resource k at time t by schedule $\omega \in \tilde{\Omega}_i$

$$\min \sum_{i \in \mathcal{P}} \sum_{\omega \in \tilde{\Omega}_i} c_{i\omega} \lambda_{i\omega}$$

minimize sum of makespans

$$s.t. \sum_{\omega \in \tilde{\Omega}_i} \lambda_{i\omega} = 1 \qquad \forall i \in \mathcal{P}$$

each project has one schedule

$$\sum_{i \in \mathcal{P}} \sum_{\omega \in \tilde{\Omega}_i} r_{i\omega kt} \lambda_{i\omega} \leq R_k^G \qquad \forall k \in \mathcal{R}^G,\ t \in \mathcal{T}$$

resource constraints

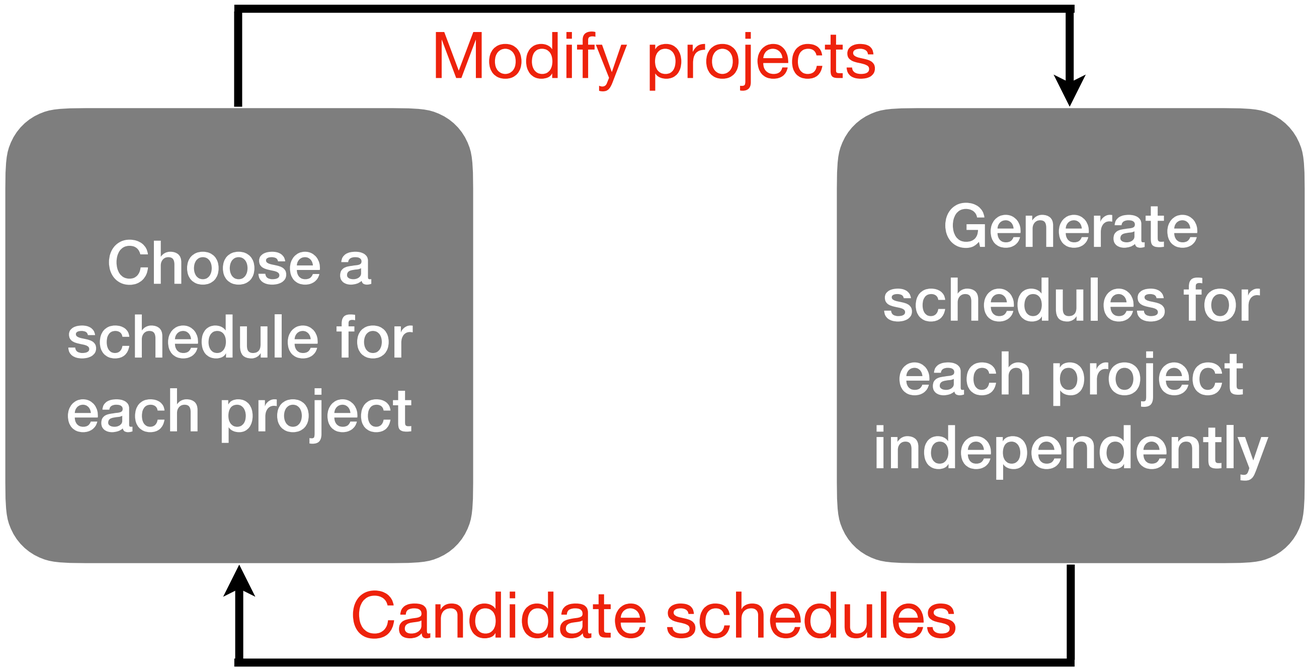
$$\lambda_{i\omega} \in \{0, 1\} \qquad \forall i \in \mathcal{P},\ \omega \in \tilde{\Omega}_i$$

Reduced Costs

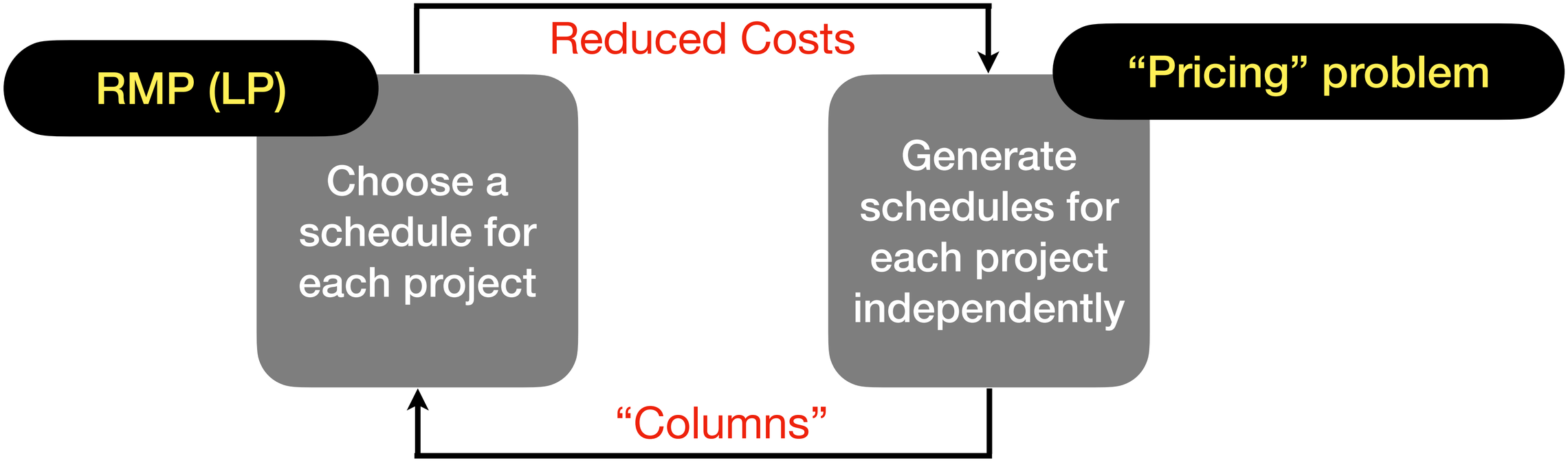
- Solving the relaxed RMP generates
  - a solution to the linear relaxation
  - "reduced costs" for each constraint
- Because of math (LP Duality Theory), reduced costs can be associated with variables and an LP solution is optimal if there do not exist variables with negative reduced costs

Implication: we can use reduced costs to generate new project solutions until we hit LP optimality

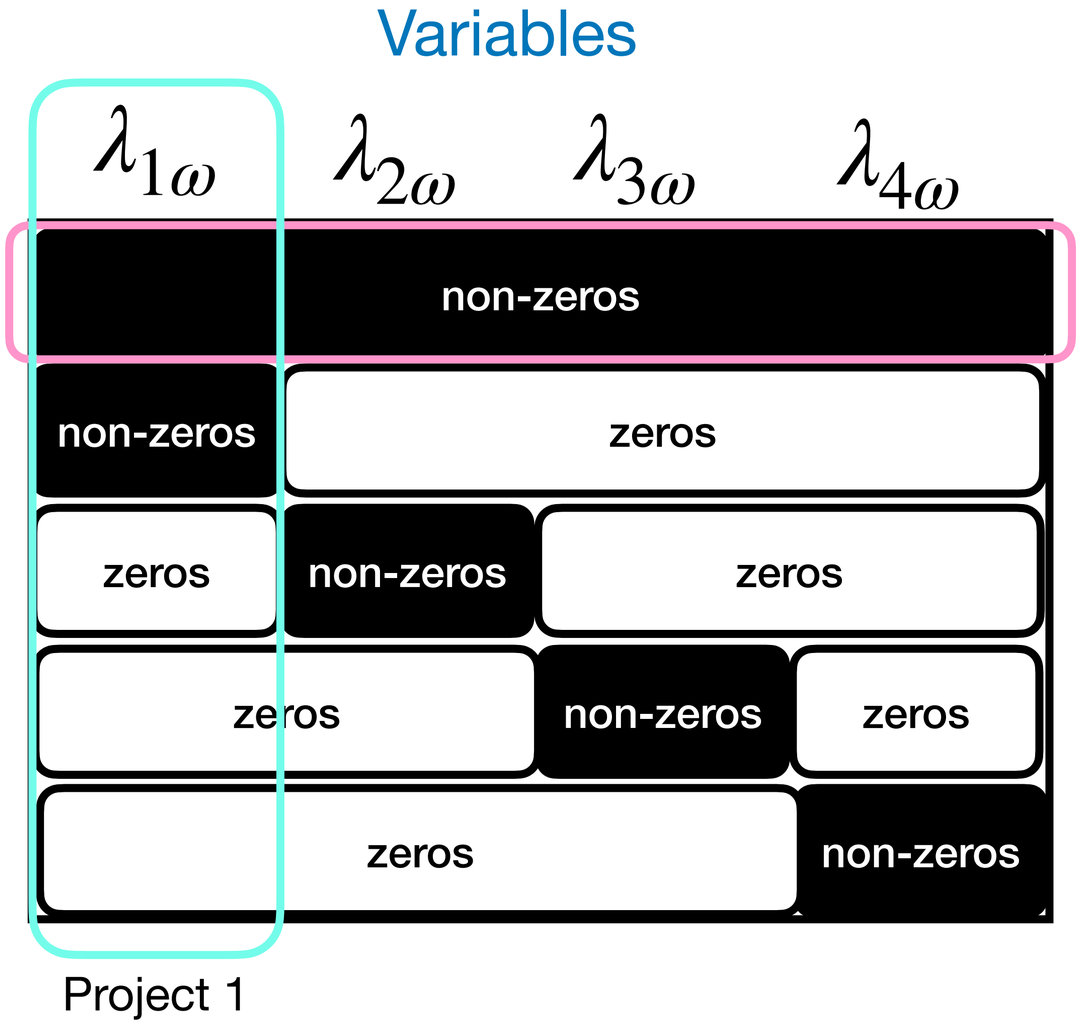
Decomposition



Decomposition



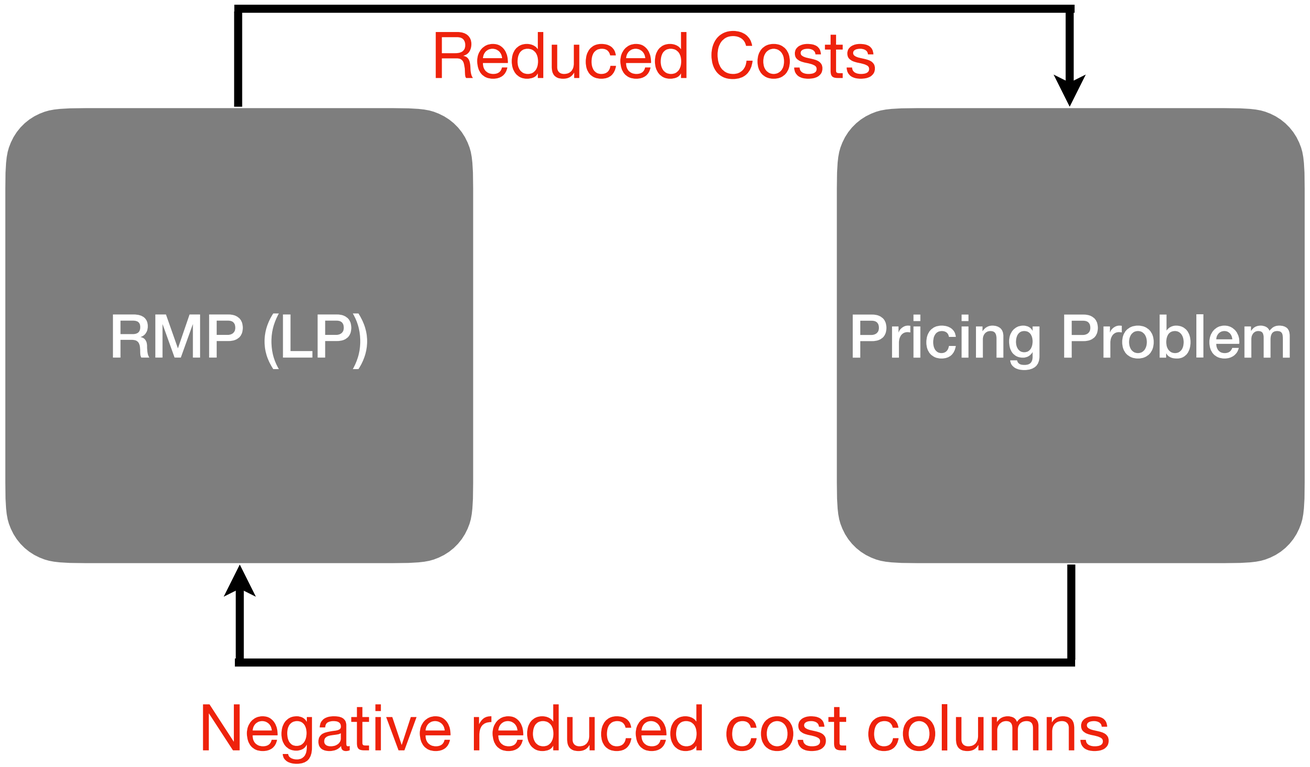
Matrix View of RMP



Each new project schedule corresponds to a column in the RMP matrix

"Column Generation"

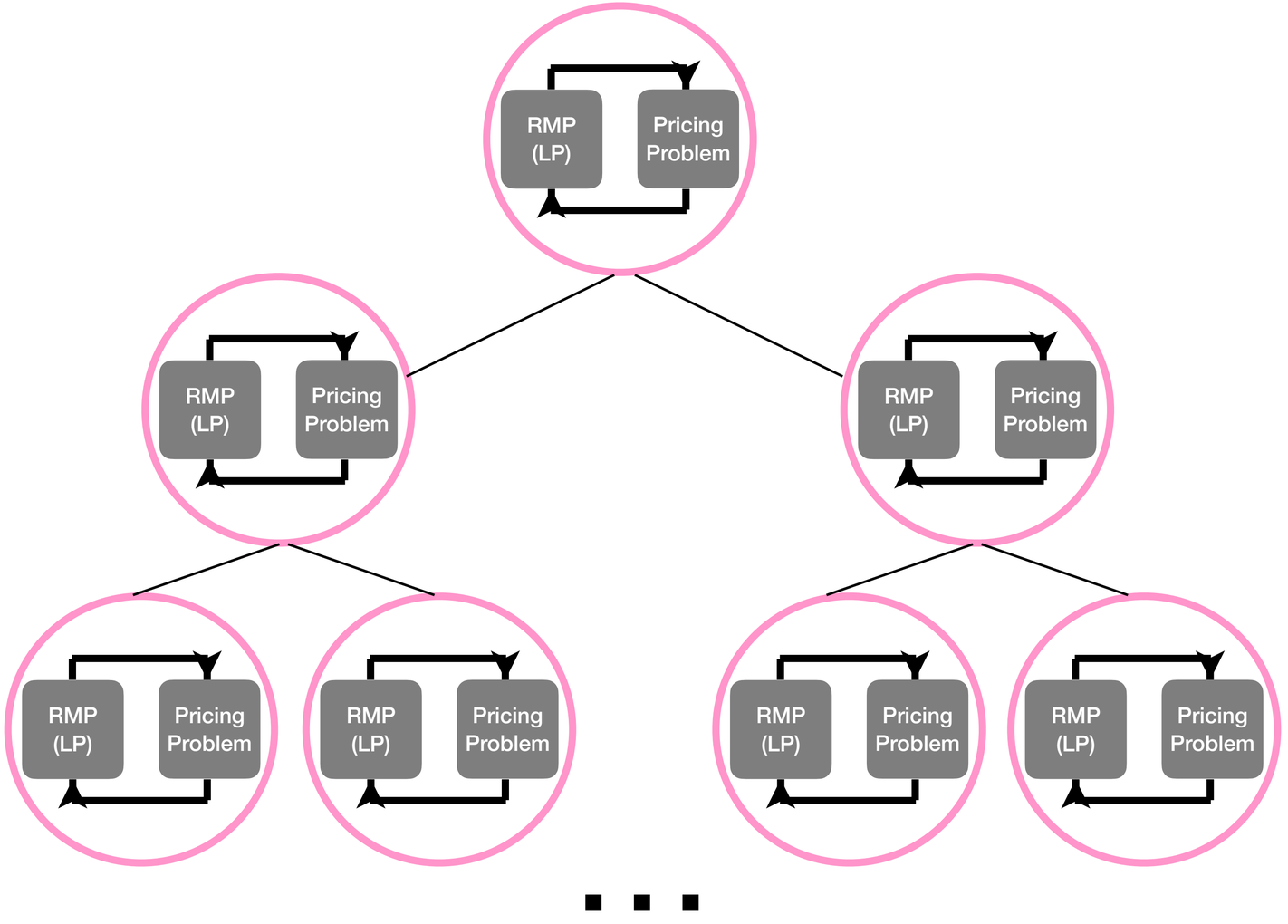
Column Generation to Solve Relaxed RMP



Solves the LP relaxation!

Branch-and-Price

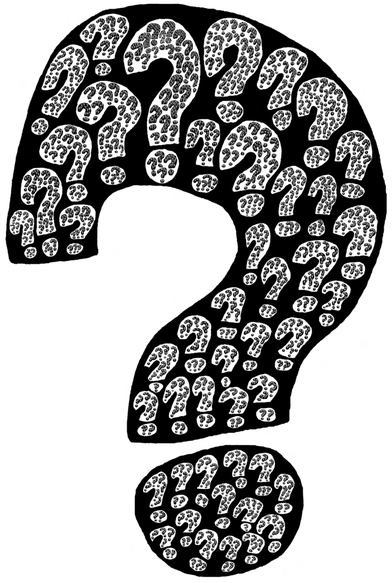
- Do column generation at each node in a branch-and-bound



Summary

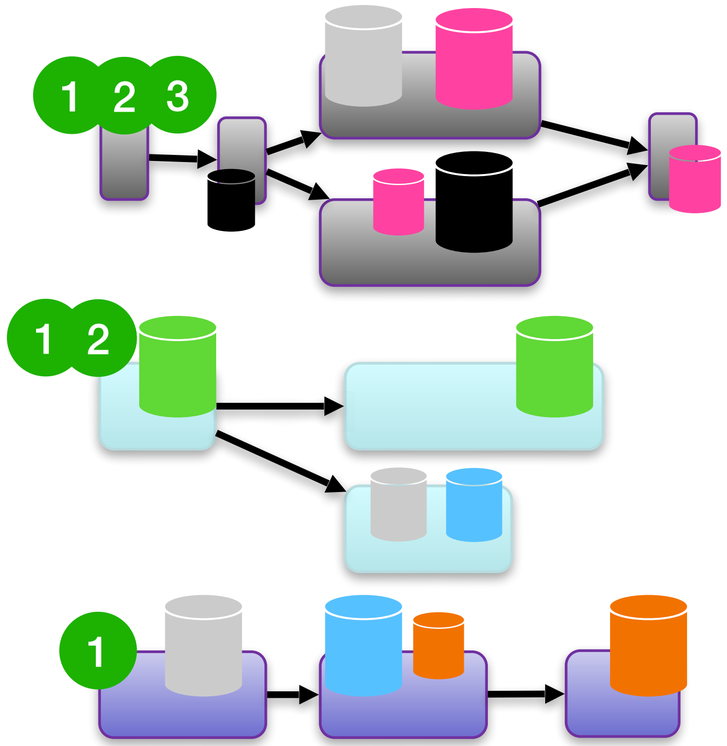
- Dantzig-Wolfe Decomposition
  - exploit “block structure” of constraints
- Column Generation
  - use duality to lazily generate variables (i.e., columns) for restricted master problem via pricing problem
- Branch-and-Price
  - do CG in each node of a search tree

Well-developed, advanced MILP technique (especially vehicle routing problems)

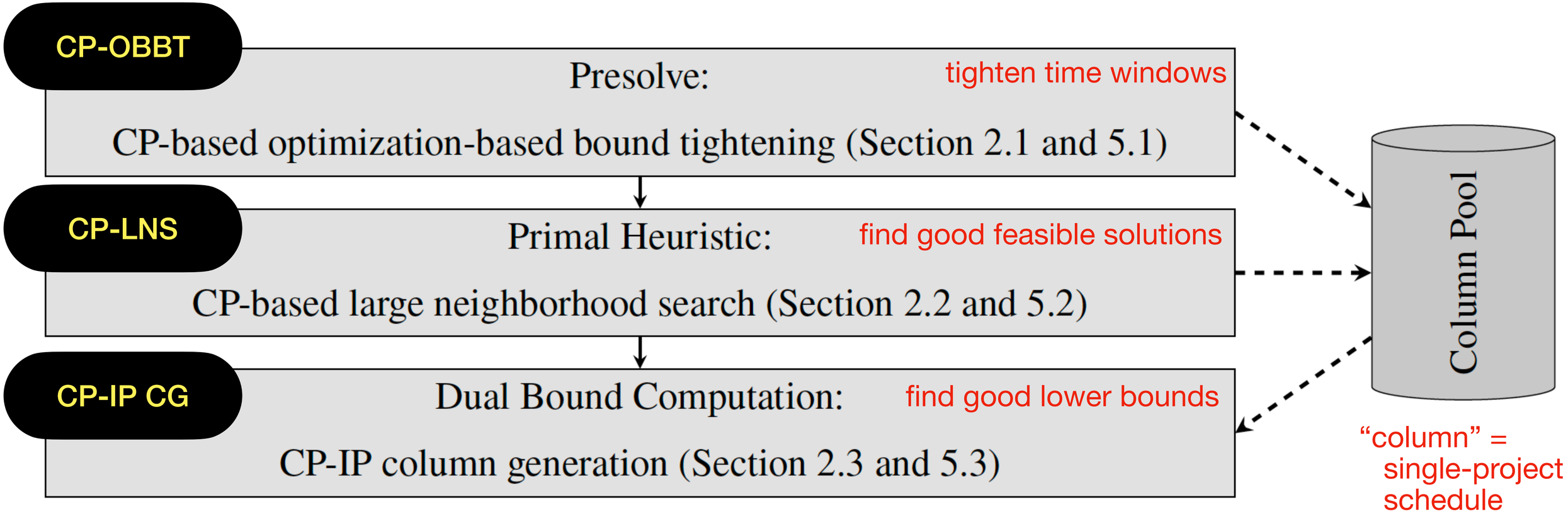


P3: MRCMPSP

- Assign a mode and a start time to each operation to respect resource capacities, to satisfy precedence constraints, and to minimize sum of project makespan (“delay”)



MILP/CP Hybrid via Dantzig-Wolfe Decomposition



[Kolter et al., *IJOC*, under review]

Large-Neighborhood Search (LNS)

- You have a CP (or MILP) model but the problem is too big/hard to solve to optimality in a reasonable time
- But you can find feasible solutions
- Idea
  - from a solution, fix a subset of variable assignments
  - discard the rest
  - solve your model on the resulting sub-space
  - if better, that is your new solution
  - repeat

[Shaw, *CP*, 1998]

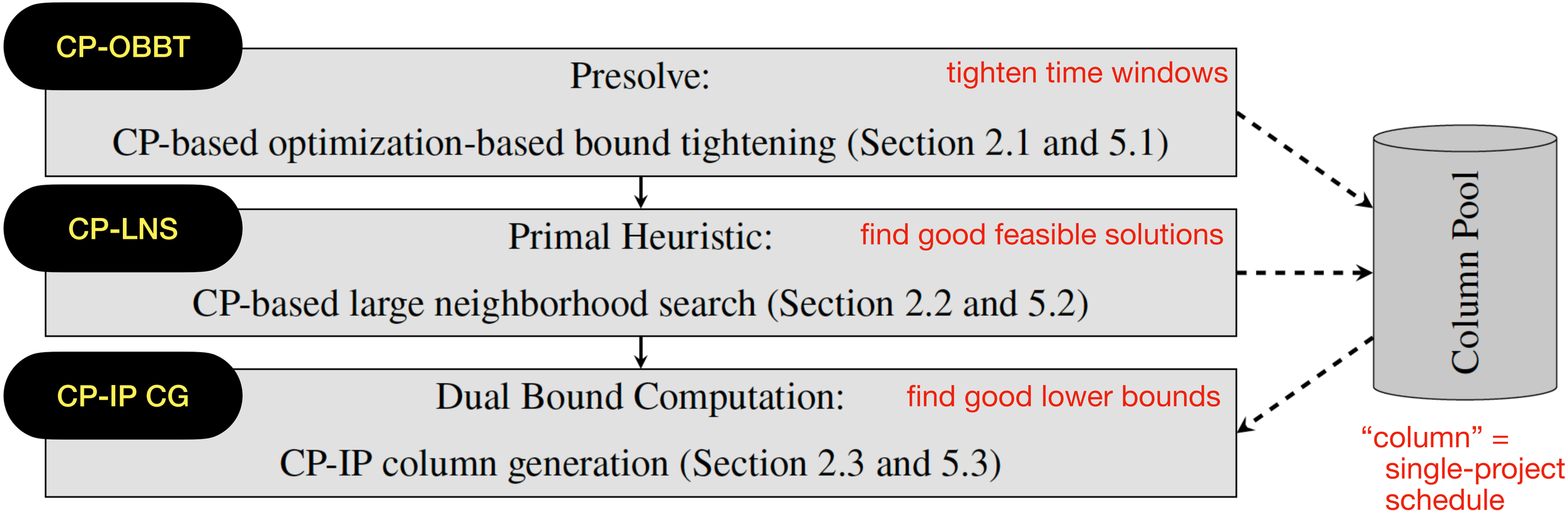
CP-LNS

- Solve monolithic (i.e., whole problem) CP model for some time limit
- LNS
  - randomly choose a subset of project schedules to discard & re-solve
    - start with small subsets (1, 2, …) and increase if no improving solution is found
  - repeat

Acceleration strategies (parallization, heuristics) explored in the paper

Every project schedule is added to the column pool

MILP/CP Hybrid via Dantzig-Wolfe Decompsition



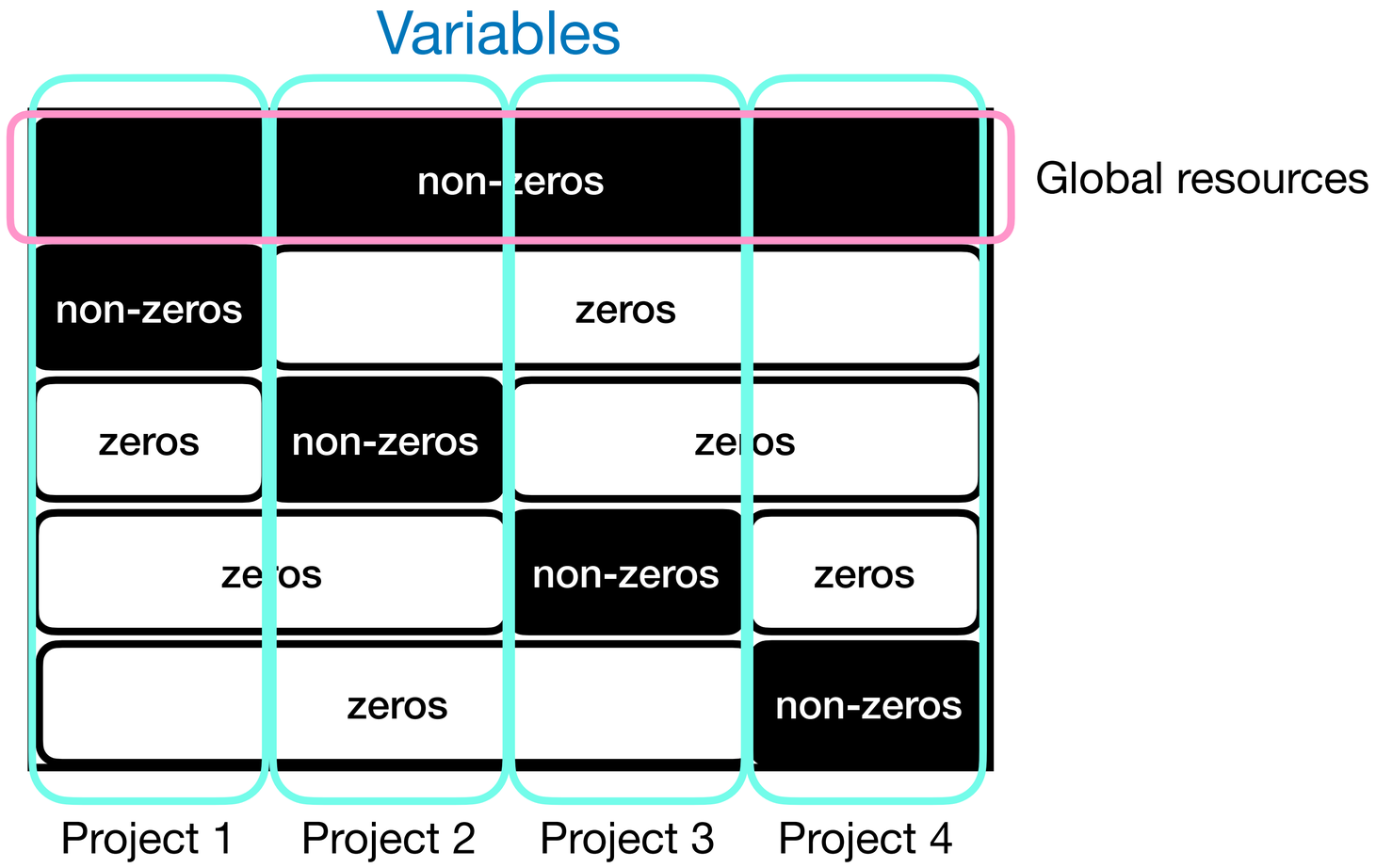
[Kolter et al., *IJOC*, under review]

Dantzig-Wolfe Decompostion

- Our observation is a general pattern that can occur

A matrix representing an IP

Linear Constraints



Dantzig-Wolfe Restricted Master Problem

$\tilde{\Omega}_i$: set of some schedules for project i

$c_{i\omega}$: makespan of schedule $\omega \in \tilde{\Omega}_i$

$r_{i\omega kt}$: usage of global resource k at time t by schedule $\omega \in \tilde{\Omega}_i$

$$\min \sum_{i \in \mathcal{P}} \sum_{\omega \in \tilde{\Omega}_i} c_{i\omega} \lambda_{i\omega}$$

minimize sum of makespans

$$s.t. \sum_{\omega \in \tilde{\Omega}_i} \lambda_{i\omega} = 1 \quad \forall i \in \mathcal{P}$$

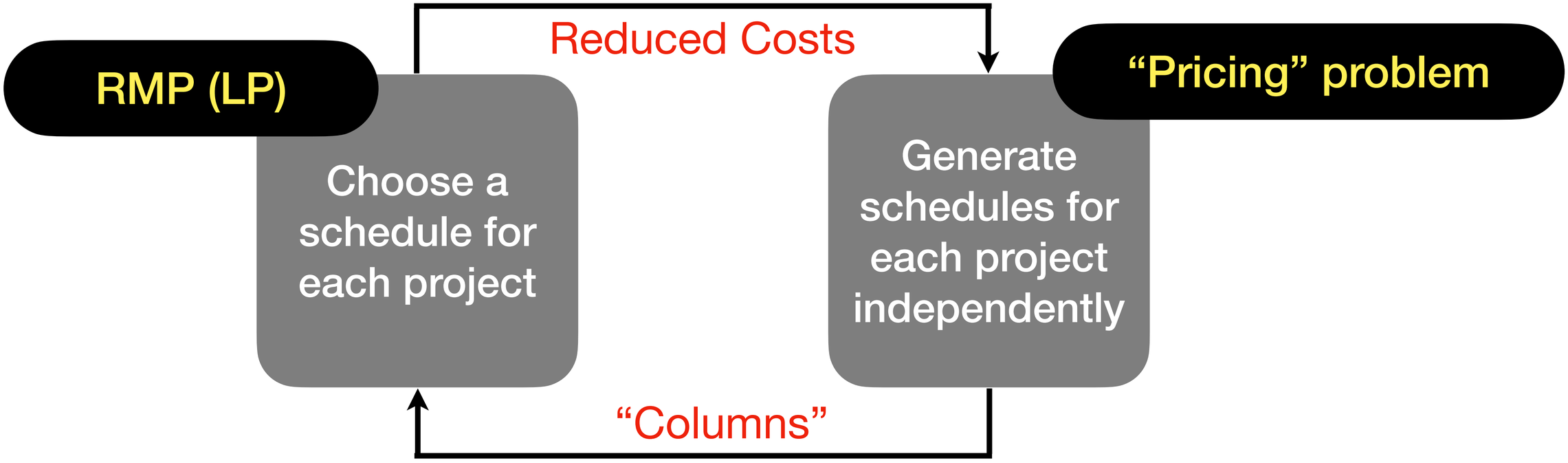
each project has one schedule

$$\sum_{i \in \mathcal{P}} \sum_{\omega \in \tilde{\Omega}_i} r_{i\omega kt} \lambda_{i\omega} \leq R_k^G \quad \forall k \in \mathcal{R}^G,\ t \in \mathcal{T}$$

resource constraints

$$\lambda_{i\omega} \in \{0, 1\} \quad \forall i \in \mathcal{P},\ \omega \in \tilde{\Omega}_i$$

Decomposition



Solving Pricing Problem

```
while CP time left
  solve RMP with IP and current column pool
  for each project
    solving pricing problem with CP
    add negative columns to column pool
while IP time left and negative columns exist
  solve RMP with IP and current column pool
  for each project
    solving pricing problem with IP
    add negative columns to column pool
  lb = max(lb, Lagrangian dual bound)
return lb
```

quickly generate good columns

use CP model projected to one project

strengthen the lower bound

use IP model projected to one project

UB, LB, and Gap

- UB (upper bound) is a feasible solution (“primal solution”)
- LB (lower bound) of k means there is no solution better than k
- If we can’t find an optimal solution, we want both a good upper bound and a quality guarantee
  - optimality gap = (UB - LB) / LB
- Critical point: UB and LB come from the same solver
- Meta-heuristics only produce UBs, no quality guarantee/ estimate

Results

Instances	Algorithm	Gap [%]	Time [s]	#Feas	#Opt	#Best		
						UB	LB	Gap
Small	IP	33.41	2,167	7	**4**	4	4	4
	CP	45.46	1,910	**8**	**4**	**6**	4	4
8 instances	CP-DWD	**12.38**	**1,608**	**8**	3	4	**7**	**7**
Medium	IP	94.77	**3,600**	7	0	0	0	0
	CP	93.51	**3,600**	**10**	0	**8**	0	0
10 instances	CP-DWD	**28.53**	**3,600**	**10**	0	1	**10**	**10**
Large	IP	97.84	**3,600**	10	0	0	**6**	0
	CP	99.10	**3,600**	**12**	0	**10**	0	0
12 instances	CP-DWD	**40.72**	**3,600**	**12**	0	2	**6**	**12**
All	IP	79.64	3,218	24	**4**	4	10	4
	CP	82.93	3,149	30	**4**	**24**	4	4
30 instances	CP-DWD	**29.10**	**3,069**	**30**	3	7	**23**	**29**

Table 1 Results for the 30 instances from the MISTA Challenge.

gap = (UB - LB) / LB

Gap reduction by 50% on average

[Kolter et al., *IJOC*, under review]

Ablation Results

Instances	Algorithm	Gap [%]	Time [s]	#Feas	#Opt	#Best		
						UB	LB	Gap
Small	No Presolve	12.82	3013.00	**8**	**4**	4	1	**5**
	No Primal Heuristic	100.00	2893.60	0	0	0	2	0
	CP+CP-IP-CG	12.66	3011.80	**8**	**4**	**5**	1	**5**
	CP-LNS+IP-CG	14.15	2922	**8**	**4**	2	1	4
8 instances	CP-DWD	**12.38**	**2572**	**8**	**4**	2	**4**	5
Medium	No Presolve	28.94	**3,600**	**10**	0	5	1	3
	No Primal Heuristic	100.00	**3,600**	0	0	0	4	0
	CP+CP-IP-CG	**28.04**	**3,600**	**10**	0	**7**	1	**5**
	CP-LNS+IP-CG	38.02	**3,600**	**10**	0	1	0	0
10 instances	CP-DWD	28.53	**3,600**	**10**	0	1	**6**	3
Large	No Presolve	41.29	**3,600**	**12**	0	5	1	3
	No Primal Heuristic	100.00	**3,600**	0	0	0	3	0
	CP+CP-IP-CG	41.33	**3,600**	**12**	0	**9**	1	4
	CP-LNS+IP-CG	63.16	**3,600**	**12**	0	3	0	0
12 instances	CP-DWD	**40.72**	**3,600**	**12**	0	4	**8**	**5**
All	No Presolve	29.58	3491	**30**	**4**	14	3	11
	No Primal Heuristic	100.00	3469	0	0	0	9	0
	CP+CP-IP-CG	29.25	3491	**30**	**4**	**21**	3	**14**
	CP-LNS+IP-CG	41.71	3474	**30**	**4**	6	1	4
30 instances	CP-DWD	**29.10**	**3410**	**30**	**4**	7	**18**	13

Table 3 Results for different algorithmic configurations.

No Presolve (no OBBT)
- little impact on gap
- better UB, worse LB: tradeoff

No Primal
- no feasible solutions

CP+CP-IP-CG (CP vs CP-LNS)
- little impact on gap
- some primal heuristic needed

CP-LNS+IP-CG (only IP pricing)
- CP pricing is important

gap = (UB - LB) / LB

[Kolter et al., *IJOC*, under review]

Results

meta-heuristics

Algorithm	Virtual Gap [%]	#Best UB
Asta et al. (2016)	17.92	16
Toffolo et al. (2016)	23.86	4
Ahmeti and Musliu (2021)	**17.40**	**18**
IP	78.59	4
CP	23.30	4
CP-DWD	25.50	3
Virtual Best	16.84	30

Table 4 **Solution quality for the 30 instances from the MISTA Challenge.**

virtual gap: uses best LB from CP/IP/CP-DWD

[Kolter et al., *IJOC*, under review]

P3: Summary

MILP

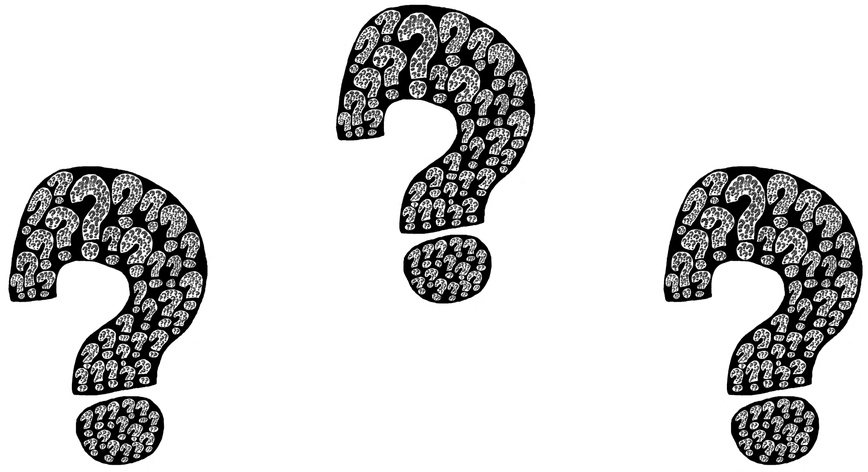
- OK lower bound
- poor upper bound

CP

- good upper bound
- poor lower bound

CP-DWD

- exploit decomposition
- upper bound almost as good as CP
- lower bound way better than anything else



“A 3-Hour Tour”

Scheduling is the allocation of resources to operations over time

Scheduling vs. Planning

Scheduling & Complexity

Solving: MILP, CP, & DIDP

P1: $1 \mid r_j, inv \mid C_{max}$

P2: SUALBP

P3: MRCMPSP

Column Generation, Dantzig-Wolfe

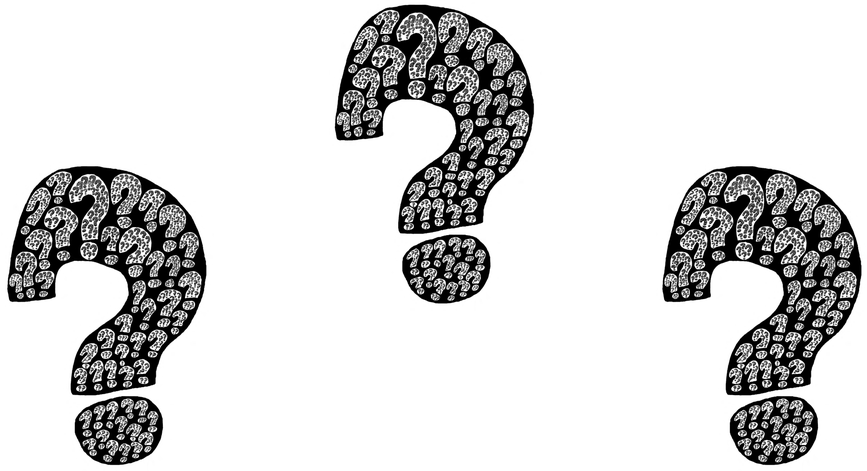
UNIVERSITY OF TORONTO | Engineering

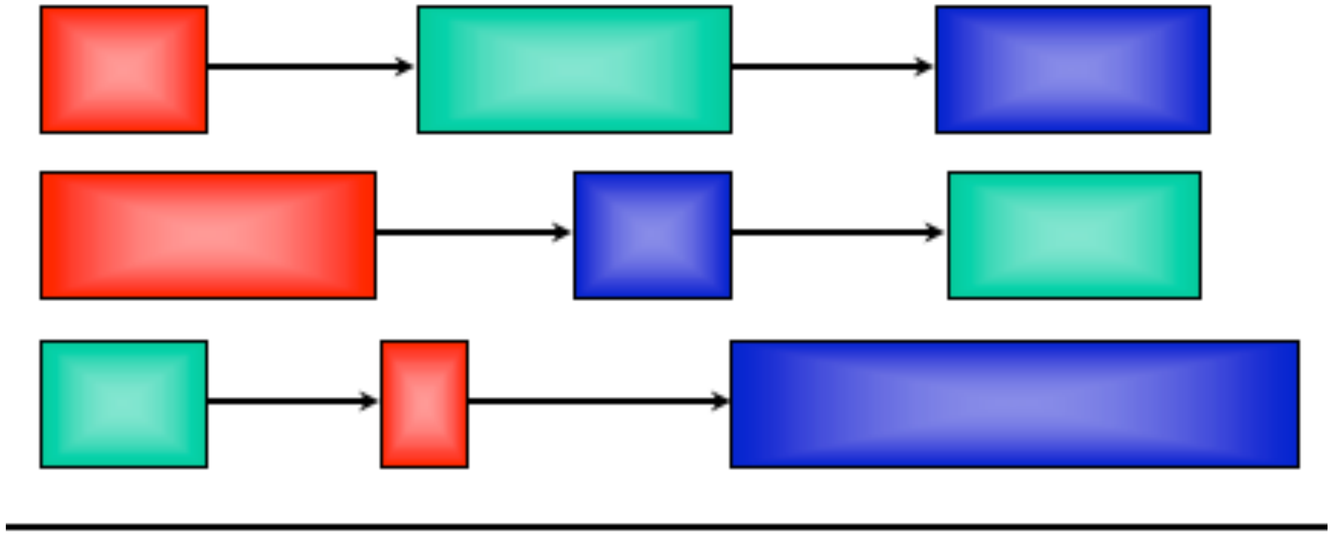
icaps Summer School 2026

Putting the S in ICAPS

A Tutorial on Scheduling

J. Christopher Beck
Department of Mechanical & Industrial Engineering
University of Toronto
chris.beck@utoronto.ca





Bonus Content

By Muhammad Mahdi Karim Facebook
The making of this document was supported by Wikimedia CH.
Own work, GFDL 1.2, https://commons.wikimedia.org/w/index.php?curid=15925090

Scheduling and ML

- End-to-end ML
  - Good for heuristics
  - I’m skeptical about optimality and complex constraints

- ML to Guide Solvers
  - Established work in MILP (e.g., Elias Khalil’s work)
  - CP: *JAIR* Special Track on CP & ML (jair.org)
  - Early work in DIDP [Narita & Beck, CPAIOR2025]

Scheduling and ML

- ML for modeling
  - e.g., English-to-MILP model
  - Moving fast
  - Maybe some of this work (P1?) could be done by LLM?

Clarke's First Law

- When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong

Arthur C. Clarke